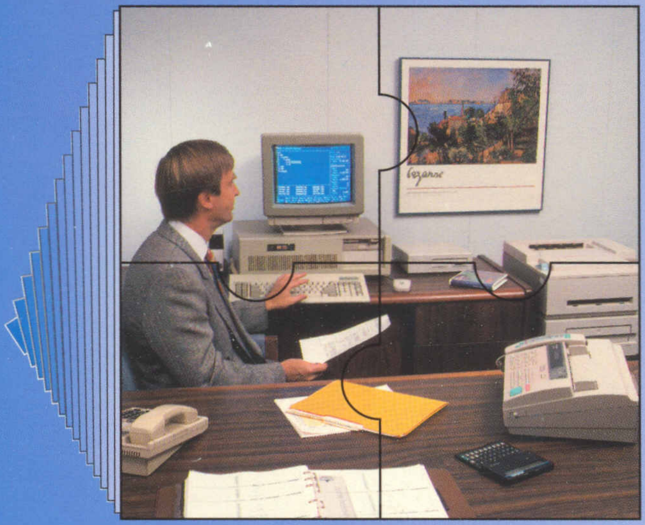
ZiLOG

ARROW ELECTRONICS, INC.
ARROW ELECTRONICS CANADA LTD.
1093 MEYERSIDE
MISSISSAUGA (ONTARIO) L5T 1M4
416-670-7769 800-387-3962 FAX: 416-670-7781

Z380™
Microprocessor Unit



**Microprocessor Solutions
for Datacommunications and
Computer Peripheral Applications**

User's Manual

ZILOG

Z380™
CPU Central Processing Unit

User's Manual

PREFACE

Thank you for your interest in the Z380™ CPU (Central Processing Unit) and its associated family of products. This Technical Manual describes programming and operation of the Z380™ Superintegration™ Core CPU, which is found in the Z380 MPU (Microprocessor Processing Unit), and future products built around Z380™ CPU core. For the external interface and detailed descriptions of the on-chip peripherals for each Superintegration device, please refer to individual product specifications.

This Technical manual consists of the following Sections:

1. **Z380™ Architectural Overview**
   Chapter 1 is an introductory section covering the key features and giving an overview of the architecture of the device.

2. **Address Spaces**
   Chapter 2 explains the address spaces the Z380 CPU can handle. Also, this chapter includes a brief description of the on-chip registers.

3. **Native/Extended Mode, Word/Long Word Mode of Operation, and Decoder Directives**
   This chapter provides a detailed explanation on the Z380's unique features, operation modes, and the Decoder Directives.

4. **Addressing Modes and Data Types**
   Chapter 4 describes the Addressing mode and data types which the Z380 can handle.

5. **Instruction Set**
   Chapter 5 contains an overview of the instruction set; as well as a detailed instruction-by-instruction description in alphabetical order.

6. **Interrupts and Traps**
   Chapter 6 explains the interrupts and traps features of the Z380.

7. **Reset**
   Chapter 7 describes the Reset function.

Appendix A

Appendix A covers the Z380's instruction format.

Appendix B

Appendix B contains all Z380 instructions sorted in Alphabetical Order.

Appendix C

Appendix C contains all Z380 instructions sorted in Numerical Order.

Appendix D

The Tables in Appendix D lists all the Z380 instructions in instruction affected by Native/Extended mode and Word/Long Word mode.

Appendix E

The Tables in Appendix E lists all the Z380 instructions in instruction affected by DDIR IM (Immediate Decoder Directives) mode.

Index

A to Z lising of Z380™ User's Manual key words and phrases.

Superintegration™ Products Guide

Description of product offerings by market niche.

Literature Guide and Ordering Information

A complete list of Zilog's literature.

This manual assumes the reader has a basic knowledge of CPU based system architectures and software development systems, such as the use of the text editor, and invoking the assembler/ compiler. Also, knowledge of the Z80 CPU architecture is desirable.

ZiLOG

Table of Contents

Z380™ Architectural Overview 1

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

ZILOG

TABLE OF CONTENTS

Chapter 1 Z380™ Architectural Overview

1.1 Introduction 1-1
1.2 CPU Architecture 1-3
  1.2.1 Modes of Operation 1-2
    1.2.1.1 Native Mode and Extended Mode 1-3
    1.2.1.2 Word or Long Word Mode 1-3
  1.2.2 Address Spaces 1-3
  1.2.3 Data Types 1-4
  1.2.4 Addressing Modes 1-4
  1.2.5 Instruction Set 1-4
  1.2.6 Exception Conditions 1-4
1.3 Benefits of the Architecture 1-5
  1.3.1 High Throughput 1-5
  1.3.2 Linear Memory Address Space 1-5
  1.3.3 Enhanced Instruction Set with 16-Bit and 32-Bit Manipulation Capability 1-5
  1.3.4 Faster Context Switching 1-6
1.4 Summary 1-6

Chapter 2 Address Spaces

2.1 Introduction 2-1
2.2 CPU Register Space 2-1
  2.2.1 Primary and Working Registers 2-3
  2.2.2 Index Registers 2-3
  2.2.3 Interrupt Register 2-3
  2.2.4 Program Counter 2-3
  2.2.5 R Register 2-3
  2.2.6 Stack Pointer 2-3
2.3 CPU Control Register Space 2-4
2.4 Memory Address Space 2-5
2.5 External I/O Address Space 2-6
2.6 On-Chip I/O Address Space 2-6

Chapter 3 Native/Extended Mode, Word/Long Word Mode of Operations and Decoder Directives

3.1 Introduction 3-1
3.2 Decoder Directives 3-2
3.3 Native Mode and Extended Mode 3-2
3.4 Word and Long Word Mode of Operation 3-3

Chapter 4 Addressing Modes and Data Types
4.1 Instruction 4-1
4.2 Addressing Mode Descriptions 4-1
  4.2.1 Register (R, RX) 4-1
  4.2.2 Immediate (IM) 4-1
  4.2.3 Indirect Register (IR) 4-2
  4.2.4 Direct Address (DA) 4-3
  4.2.5 Indexed (X) 4-4
  4.2.6 Program Counter Relative Mode (RA)
  4.2.7 Stack Pointer Relative Mode (SR) 4-6
4.3 Data Types 4-8

Chapter 5 Instruction Set
5.1 Introduction 5-1
5.2 Processor Flags 5-2
  5.2.1 Carry Flag (C) 5-3
  5.2.2 Add/Subtract Flag (N) 5-3
  5.2.3 Parity/Overflow (P/V) 5-3
  5.2.4 Half-Carry Flag (H) 5-3
  5.2.5 Zero Flag (Z) 5-3
  5.2.6 Sign Flag (S) 5-3
  5.2.7 Condition Codes 5-3
5.3 Select Register 5-4
  5.3.1 IY Bank Select (IYBANK) 5-4
  5.3.2 IY or IY' Register Select (IY') 5-4
  5.3.3 IX Bank Select (IXBANK) 5-4
  5.3.4 IX or IX' Register Select (IX') 5-4
  5.3.5 Main Bank Select (MAINBANK) 5-4
  5.3.6 BC/DE/HL or BC'/DE'/HL' Register Select (ALT) 5-4
  5.3.7 Extended Mode (XM) 5-4
  5.3.8 Long Word Mode (LW) 5-4
  5.3.9 Interrupt Enable Flag (IEF) 5-5
  5.3.10 Interrupt Mode (IM) 5-5
  5.3.11 Lock (LCK) 5-5
  5.3.12 AF or AF' Register Select (AF') 5-5
5.4 Instruction Execution and Exceptions 5.5
  5.4.1 Instruction Execution and Interrupts 5-5
  5.4.2 Instruction Execution and Trap 5-5
5.5 Instruction Set Functional Groups 5-7
  5.5.1 8-Bit Load/Exchange Group 5-6
  5.5.2 16-Bit and 32-Bit Load, Exchange, SWAP and PUSH/POP Group 5-7
  5.5.3 Block Transfer and Search Group 5-8
  5.5.4 8-Bit Arithmetic and Logical Group 5-9
  5.5.5 16-Bit Arithmetic Operation 5-10
  5.5.6 8-Bit Manipulation, Rotate and Shift Group 5-11
  5.5.7 16-Bit Manipulation, Rotate and Shift Group 5-11
  5.5.8 Program Control Group 5-12
  5.5.9 External Input/Output Instruction Group 5-13
  5.5.10 Internal I/O Instruction Group 5-15
  5.5.11 CPU Control Group 5-16

5.5.12 Decoder Directives 5-17
5.6 Notation and Binary Encoding 5-17
5.7 Execution Time 5-18

Chapter 6 Interrupts and Traps

6.1 Introduction 6-1
6.2 Interrupts 6-2
6.2.1 Interrupt Priority Ranking 6-2
6.2.2 Interrupt Control 6-2
6.2.2.1 IEF1, IEF2 6-3
6.2.2.2 I, I Extend 6-3
6.2.2.3 Interrupt Enable Register 6-3
6.2.2.4 Assigned Vectors Base Register 6-3
6.2.2.5 Trap and Break Register 6-4
6.3 Trap Interrupt 6-5
6.4 Nonmaskable Interrupt 6-5
6.5 Interrupt Response for Maskable Interrupt on /INT0 6-5
6.5.1 Interrupt Mode 0 Response for Maskable Interrupt /INT0 6-5
6.5.2 Interrupt Mode 1 Response for Maskable Interrupt /INTO 6-5
6.5.3 Interrupt Mode 2 Response for Maskable Interrupt /INT0 6-5
6.5.4 Interrupt Mode 3 Response for Maskable Interrupt /INT0 6-5
6.6 Assigned Interrupt Vectors Mode for Maskable Interrupts /INT3-/INT1 6-6
6.7 RETI Instruction 6-6

Chapter 7 Reset

7.1 Introduction 7-1

Appendix A

Z380™ CPU Instruction Formats A-1

Appendix B

Z380™ Instructions in Alphabetic Order B-1

Appendix C

Z380™ Instruction in Numeric Order C-1

Appendix D

Instructions Affected by Native/Extended Mode, and Long Word Mode D-1

Appendix E

Instructions Affected by DDIR IM Instructions E-1

Index

A through Z I-1

Superintegration™

Products Guide S-1

Literature Guide

Ordering Information L-1

FIGURES

Chapter 1
Figure 1-1. Z380™ CPU Register Architecture 1-2

Chapter 2
Figure 2-1. Register File Organization (Z380™ MPU) 2-2
Figure 2-2. Bit/Byte Ordering Conventions 2-5

Chapter 3
Figure 3-1. Z380™ CPU Operation Modes 3-1

Chapter 5
Figure 5-1. Flag Register 5-3
Figure 5-2. Select Register

Chapter 6
Figure 6-1. Interrupt Enable Register 6-3
Figure 6-2. Assigned Vectors Base Register 6-3
Figure 6-3. Trap and Break Register 6-4

TABLES

Chapter 2

Table 2.1 I/O Addressing Options 2-6

Chapter 5

Table 5-1. Condition Codes 5-1
Table 5-2. 8-Bit Load Group Instructions 5-1
Table 5-3. 8-Bit Load Group Allowed Source/Destination Combinations 5-1
Table 5-4. 16-Bit and 32-Bit load, Exchange, PUSH/POP Group Instructions 5-1
Table 5-5. Supported Sours and Destination Combination for 16-Bit and 32-Bit Exchange Instructions .. 5-1
Table 5-6. Supported Source and Destination Combination for 16-Bit and 32-Bit Load Instructions 5-1
Table 5-7. Supported Operand for PUSH/POP Instructions 5-2
Table 5-8. Block Transfer and Search Group 5-2
Table 5-9. Supported Source/Destination for 8-Bit Arithmetic and Logic Group 5-4
Table 5-10. 16-Bit Arithmetic Operation 5-4
Table 5-11. Bit Set/Reset/Test, Rotate and Shift Group 5-4
Table 5-12. 16-Bit Rotate and Shift Group 5-5
Table 5-13. Program Control Group Instructions 5-6
Table 5-14. External I/O Group Instructions 5-14
Table 5-15. Internal I/O Instruction Group 5-15
Table 5-16. CPU Control Group 5-16
Table 5-17. Decoder Directive Instructions 5-17
Table 5-18. Execution Time 5-18

Chapter 6

Table 6-1. Interrupt Priority Ranking 6-1
Table 6-2. Interrupt Flags and Registers 6-1
Table 6-3. Operation Effects on IEF1 and IEF2 6-1
Table 6-4. Assigned Interrupt Vectors 6-1

Chapter 7

Table 7-1. Effect of a Reset on Z380™ CPU and Related I/O Registers 7-2

Appendix A

Table A-1. Format 1 Instructions Encodings A-2
Table A-2. Format 2 Instructions Encodings A-2
Table A-3. Format 3 Instructions Encodings A-2
Table A-4. Format 4 Instructions Encodings A-2

Appendix E

Table E-1. Valid with DDIR IB in EM E-1
Table E-2. Valid with DDIR IB E-1
Table E-3. Valid with DDIR IB in LW E-1
Table E-4. Valid with DDIR IB E-2
Table E-5. Valid with DDIR IW in EM E-2
Table E-6. Valid with DDIR IW E-4
Table E-7. Valid with DDIR IW in LW E-5
Table E-8. Valid with DDIR IW E-5

Table of Contents

Z380™ Architectural Overview **1**

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

CHAPTER 1

Z380™ ARCHITECTURAL OVERVIEW

1.1 INTRODUCTION

Zilog's new Z380™ Central Processing Unit (CPU) is a high-performance CPU engine designed to meet today's application requirements. The Z380 CPU incorporates advanced architectural features that allow fast and efficient throughput and increased memory addressing capabilities while maintaining Z80® CPU and Z180® MPU object-code compatibility. The Z380 CPU core offers a continuing growth path for present Z80 or Z180® based designs and serves as a high-performance CPU engine for successive generations of Superintegration™ devices. The first implementation of this CPU is the Z380 MPU. It has the following key features:

- Full Static CMOS Design with Low Power Standby Mode Support
- DC to 40 MHz Operating Frequency
- Enhanced Instruction Set that Maintains Object-Code Compatibility with Z80 and Z180 Microprocessors
- 16-Bit (64K) or 32-Bit (4G) Linear Address Space
- 16-Bit Internal Data Bus
- Two Clock Cycle Instruction Execution Minimum
- Multiple On-Chip Register Files (Z380 MPU has Four Banks)
- BC/DE/HL/IX/IY Registers are Augmented by 16-Bit Extended Registers (BCz/DEz/HLz/IXz/IYz), PC/SP/I Registers are Augmented by Extended Registers (PCz/SPz/Iz) for 32-Bit Addressing Capability.
- Newly Added IX' and IY' Registers with Extended Registers (IXz'/IYz')
- Enhanced Interrupt Capabilities, Including 16-Bit Vector
- Undefined Opcode Trap for Full Z380 CPU Instruction Set

The Z380 CPU is an enhanced version of the Z80 CPU. The Z80 CPU instruction set has been retained, meaning that the Z380 CPU is completely binary-code compatible with present Z80 and Z180 codes. The basic addressing modes of the Z80 microprocessor have been augmented with Stack Pointer Relative loads and stores, 16-bit and 24-bit Indexed offsets, and more flexible Indirect register addressing, with all of the addressing modes allowing access to the entire 32-bit address space. Significant additions have been made to the instruction set with a full complement of 16-bit arithmetic and logical operations, 16-bit I/O operations, multiply and divide, a complete set of register-to-register loads and exchanges, plus 32-bit load and exchange, and 32-bit arithmetic operation for address calculation.

The basic register file of the Z80 microprocessor is expanded to include alternate register versions of the IX and IY registers. There are four sets of this basic Z80 microprocessor register file present in the Z380 MPU, along with the necessary resources to manage switching between the different register sets. All of the register pairs and index registers in the basic Z80 microprocessor register file are expanded to 32 bits.

The Z380 CPU expands the basic 64 Kbyte Z80 and Z180 address space to a full 4 Gbyte (32-bit) address space. This address space is linear and completely accessible to the user program. The external I/O address space is similarly expanded to a full 4 Gbyte (32-bit) range, and 16-bit I/O, both simple and block move, are included. A 256 byte-wide internal I/O space is newly added. This space is used to access on-chip I/O resources on the future Superintegration™ implementation of this CPU core.

For a detailed description of the basic register architecture of the Z380 CPU, see Figure 1-1. Note that this is the implementation of the Z380 MPU, and the size of the register banks are four each, but the Z380 CPU architecture allows expansion of up to 128 sets of each.

1.1 INTRODUCTION (Continued)

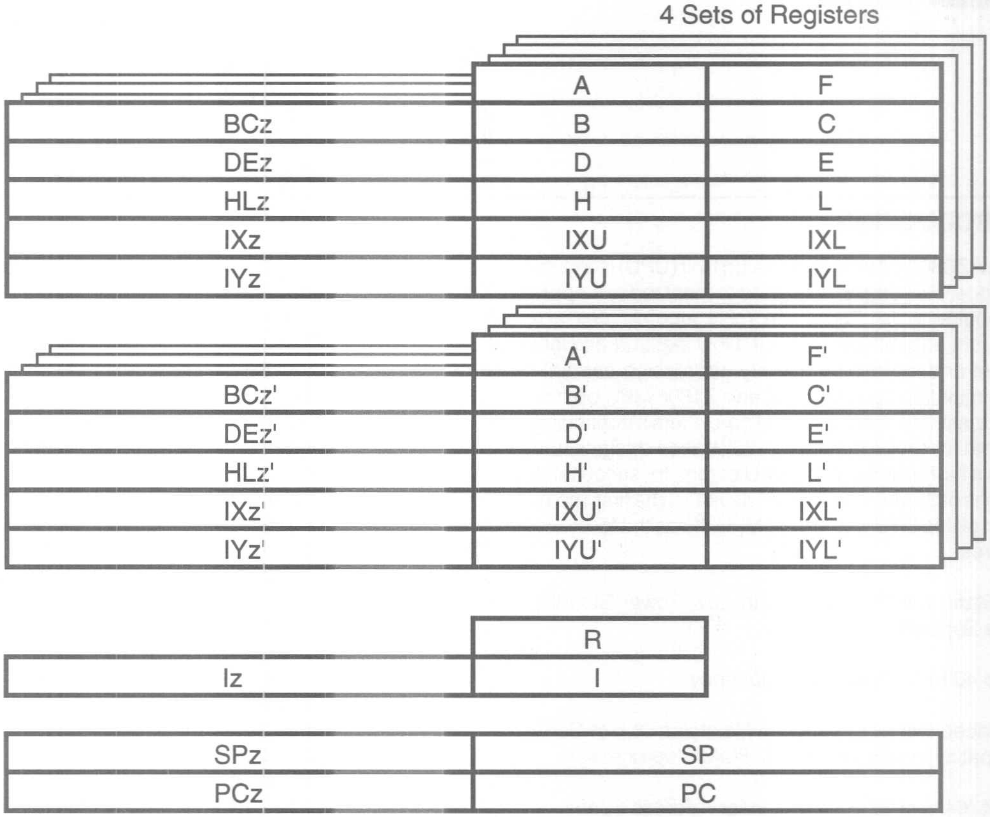

4 Sets of Registers
A
F
BCz
B
C
DEz
D
E
HLz
H
L
IXz
IXU
IXL
IYz
IYU
IYL
A'
F'
BCz'
B'
C'
DEz'
D'
E'
HLz'
H'
L'
IXz'
IXU'
IXL'
IYz'
IYU'
IYL'
R
Iz
I
SPz
SP
PCz
PC

Figure 1-1. Z380™ CPU Register Architecture

1.2 CPU ARCHITECTURE

The Z380 CPU is a binary-compatible extension of the Z80 CPU and the Z180 CPU architecture. High throughput rates are achieved by a high clock rate, high bus bandwidth, and instruction fetch/execute overlap. Communicating to the external world through an 8-bit or 16-bit data bus, the Z380 CPU is a full 32-bit machine internally, with a 32-bit ALU and 32-bit registers.

1.2.1 Modes of Operation

To maintain compatibility with the Z80/Z180 CPU while having the capability to manipulate 4 Gbytes of memory address range, the Z380 CPU has two bits in the Select Register (SR) to control the modes of operation. One bit controls the address manipulation mode: Native mode or Extended mode; and the other bit controls the data manipulation mode: Word mode or Long Word mode. In result, the Z380 CPU has four modes of operation. On reset, the Z380 CPU is in Native/Word mode, which is compatible to the Z80/Z180's operation mode. For details on this subject, refer to Chapter 3, "Native/Extended Mode, Word/Long Word Mode of Operation, and Decoder Directive Instructions."

1.2.1.1 Native Mode and Extended Mode

The Z380 CPU can operate in either Native or Extended mode, as controlled by a bit in the Select Register (SR). In Native mode (the Reset configuration), all address manipulations are performed modulo 65536 (2^{16}). In this mode, the Program Counter (PC) only increments across 16 bits, all address manipulation instructions (increment, decrement, add, subtract, indexed, stack relative, and PC relative) only operate on 16 bits, and the Stack Pointer (SP) only increments and decrements across 16 bits. The PC high-order word is left at all zeros, as the high-order words of the SP and the I register. Thus, Native mode is fully compatible with the Z80 CPU's 64 Kbyte address mode. It is still possible to address memory outside of 64 Kbyte address space for data storage and retrieval in Native mode, however, since direct addresses, indirect addresses, and the high-order word of the SP, I, and the IX and IY registers may be loaded with non-zero values. Executed code and interrupt service routines must reside in the lowest 64 Kbytes of the address space.

In Extended mode, however, all address manipulation instructions operate on 32 bits, allowing access to the entire 4 Gbyte address space of the Z380 CPU. In both Native and Extended modes, the Z380 drives all 32 bits of the address onto the external address bus; only the width of the manipulated addresses distinguishes Native from Extended mode. The Z380 CPU implements one instruction to allow switching from Native to Extended mode (SETC XM); however, once in Extended mode, only Reset will return the Z380 CPU to Native mode. This restriction applies because of the possibility of "misplacing" interrupt service routines or vector tables during the transition from Extended mode back to Native mode.

1.2.1.2 Word or Long Word Mode

In addition to Native and Extended mode, which are specific to memory space addressing, the Z380 CPU can operate in either Word or Long Word mode specific to data load and exchange operations. In Word mode (the Reset configuration), all word load and exchange operations manipulate 16-bit quantities. For example, only the low-order words of the source and destination are exchanged in an exchange operation, with the high-order words unaffected.

In the Long Word mode, all 32 bits of the source and destination are exchanged. The Z380 CPU implements two instructions plus decoder directives to allow switching between Word and Long Word mode; SETC LW (Set Control Long Word) and RESC LW (Reset Control Long Word) perform a global switch, while DDIR W, DDIR LW and their variants are decoder directives that select a particular mode only for the instruction that they precede.

Note that all word data arithmetic (as opposed to address manipulation arithmetic), rotate, shift, and logical operations are always in 16-bit quantities. They are not controlled by either the Native/Extended or Word/Long Word selections. The exceptions to the 16-bit quantities are, of course, those multiply and divide operations with 32-bit products or dividends.

All word Input/Output operations are performed on 16-bit values, regardless of Word/Long Word operation.

1.2.2 Address Spaces

Addressing spaces in the Z380 CPU include the CPU register, the CPU control register, the memory address, on-chip I/O address, and the external I/O address. The CPU register space is a superset of the Z80 CPU register set, and consists of all of the registers in the CPU register file. These CPU registers are used for data and address manipulation, and are an extension of the Z80 CPU register set, with four sets of this extended Z80 CPU register set present in the Z380 CPU. Access to these registers is specified in the instruction, with the active register set selected by bits in the Select Register (SR) in the CPU control register space.

1.2.2 Address Spaces (Continued)

Each register set includes the primary registers A, F, B, C, D, E, H, L, IX, and IY, as well as the alternate registers A', F', B', C', D', E', H', L', IX', and IY'. Also, IX, IX', IY, and IY' registers are accessible as two byte registers, each named as IXU, IXL, IXU' IXL', IYU, IYL, IYU', and IYL'. These byte registers can be paired B with C, D with E, H with L, B' with C', D' with E', and H' with L' to form word registers, and these word registers are extended to 32 bits with the "z" extension to the register. This register extension is only accessible when using the register as a 32-bit register (in the Long Word mode) or when swapping between the most-significant and least-significant word of a 32-bit register using SWAP instructions. Whenever an instruction refers to a word register, the implicit size is controlled by Word or Long Word mode. Also included are the R, I, and SP registers, as well as the PC.

The Select Register (SR) determines the operation of the Z380 CPU. The contents of this register determine the CPU operating mode, which register bank will be used, the interrupt mode in effect, and so on.

The Z380 CPU's memory address space is linear 4 Gbytes. To keep compatibility with the Z80 CPU memory addressing model, it has two control bits to change its operation modes—Native or Extended, Word or Long Word.

The Z380 CPU architecture also distinguishes between the memory and I/O addressing space and, therefore, requires specific I/O instructions. Furthermore, I/O addressing space is subdivided into the on-chip I/O address space and the external I/O addressing space. External I/O addressing space in the Z380 CPU is 32 bits long, and internal I/O addressing space is 8-bits long. There are separate sets of I/O instructions for each I/O addressing space.

Some of the Internal I/O registers are used to control the functionality of the device, such as to program/read status of Trap, Assigned Vector Base address, enabling of interrupts, and to get Chip version ID.

For details on this topic, refer to Chapter 2, "Address Spaces."

1.2.3 Data Types

Many data types are supported by the Z380 CPU architecture. The basic data type is the 8-bit byte, which is also the basic addressable memory element. The architecture also supports operations on bits, BCD (Binary Coded Decimal) digits, words (16 bits or 32 bits), byte strings and word strings. For details on this topic, refer to Section 4.3, "Data Types."

1.2.4. Addressing Modes

Addressing modes are used by the Z380 CPU to calculate the effective address of an operand needed for execution of an instruction. Seven addressing modes are supported by the Z380 CPU. Of these seven, one is an addition to the Z80 CPU addressing modes (Stack Pointer Relative) and the remaining six modes are either existing or extensions to Z80 CPU addressing modes.

- Register
- Immediate
- Indirect Register
- Direct Address
- Indexed
- Program Counter Relative
- Stack Pointer Relative

All addressing modes are available on the 8-bit load, arithmetic, and logical instructions; the 8-bit shift, rotate, and bit manipulation instructions are limited to the registers and Indirect register addressing modes. The 16-bit loads on the addressing registers support all addressing modes except Index, while other 16-bit operations are limited to the Register, Immediate, Indirect Register, Index, Direct Address, and PC Relative addressing modes.

For details on this subject, refer to Chapter 4, "Addressing Modes and Data Types."

1.2.5. Instruction Set

The Z380 CPU instruction set is an expansion of the Z80 instruction set; the enhancements include support for additional addressing modes for the Z80 instructions as well as the addition of new instructions. The Z380 CPU instruction set provides a full complement of 8-bit, 16-bit, and 32-bit operation, including multiplication and division.

For details on this subject, refer to Chapter 5, "Instruction Set."

1.2.6 Exception Conditions

The Z380 CPU supports three types of exceptions (conditions that alter the normal flow of program execution); interrupts, traps, and resets.

Interrupts are asynchronous events typically triggered by peripherals requiring attention. The Z380 CPU interrupt structure has been significantly enhanced by increasing the number of interrupt request lines and by adding an efficient means for handling nested interrupts. The Z380 CPU has five interrupt lines. These are: Nonmaskable Interrupt line (/NMI) and Maskable interrupt lines (/INT0, /INT1, /INT2, and /INT3). Interrupt requests on /INT3-/INT1 are handled by a newly added interrupt handing mode, "Assigned Vectored Mode," which is a fixed vectored interrupt mode similar in interrupt handling to the Z180's interrupts from on-chip peripherals. For handling interrupt requests on the /INTO line, there are four modes available:

- 8080 compatible (Mode 0), in which the interrupting device provides the first instruction of the interrupt routine.
- Dedicated interrupts (Mode 1), in which the CPU jumps to a dedicated address when an interrupt occurs.
- Vectored interrupt mode (Mode 2), in which the interrupting peripheral device provides a vector into a table of jump address.
- Enhanced vectored interrupt mode (Mode 3), wherein the CPU expects 16-bit vector, instead of 8-bit interrupt vectors in Mode 2.

The first three modes are compatible with Z80 interrupt modes; the fourth mode provides more flexibility.

Traps are synchronous events that trigger a special CPU response when an undefined instruction is executed. It can be used to increase system reliability, or used as a "software trap instruction."

Hardware resets occur when the /RESET line is activated and override all other conditions. A /RESET causes certain CPU control registers to be initialized.

For details on this subject, refer to Chapter 6, "Interrupts and Traps."

1.3 BENEFITS OF THE ARCHITECTURE

The Z380 CPU architecture provides several significant benefits, including increased program throughput achieved by higher bus bandwidth (16-bit wide bus), reduction to two clocks/basic machine cycle (vs four clocks/cycle on the Z80 CPU), prefetch cue, access to the larger linear addressing space, enhanced instructions/new addressing mode, data/address manipulation in 16/32 bits, and faster context switching by utilizing multiple register banks.

1.3.1 High Throughput

Very high throughput rates can be achieved with the Z380 CPU, due to the basic machine cycle's reduction to two clocks/cycle from four clocks/cycle on the Z80 CPU, fine tuned four staged pipeline with prefetch cue. This well designed pipeline and prefetch cue are both totally transparent to the user, thus maximizing the efficiency of the pipeline all the time. The Z380 CPU implemented onto the Z380 MPU is configured with a 16-bit wide data bus, which doubles the bus bandwidth. These architectural features result in two clocks/instructions execution minimum, three clocks/instruction on average. The high clock rates (up to 40 MHz) achievable with this processor. Make the overall performance of the Z380 CPU more than ten times that of the Z80.

1.3.2 Linear Memory Address Space

Z380 CPU architecture has 4 Gbytes of linear memory address space. The Z80 CPU architecture allows 64 Kbytes of memory addressing space. This was more than sufficient when the Z80 CPU was first developed. But as the technology improved over time, applications started to demand more complicated processing, multitasking, faster processing, etc., with the high level language needed to develop software. As a result, 64 Kbytes of memory addressing space is not enough for some Z80 CPU based applications. In order to handle more than 64 Kbytes of memory, the Z80 CPU requires a Memory Banking scheme, or MMU (Memory Management Unit), like the Z180 MPU or Z280 MPU. These provide the overhead to access more than 64 Kbytes of memory.

The Z380 CPU architecture allows access to a full 4 Gbytes (2^{32}) of memory addressing space as well as 4 Gbytes of I/O addressing area, without using a Memory Banking scheme, or MMU.

1.3.3. Enhanced Instruction Set with 16-Bit and 32-Bit Manipulation Capability

The Z380 CPU instruction set is 100% upward compatible to the Z80 CPU instruction set; that is all the Z80 instructions have been preserved at the binary level. New instructions added to the Z380 CPU include:

- Less restricted operand source/destination combinations.
- More flexible register exchange instructions.
- Stack Pointer Relative addressing mode.

1.3.3. Enhanced Instruction Set with 16-Bit and 32-Bit Manipulation Capability (Continued)

- DDIR (Decoder Directive Instructions) to enhance addressing capability to cover 4 Gbytes of memory space, as well as data manipulation capability.
- Jump relative/Call relative instructions with 8-bit, 16-bit, or 24-bit displacement.
- Full complements of 16-bit arithmetic instructions.
- 32-bit manipulate instructions for address manipulation.

These new instructions help to compact the code, as well as shorten the program's overall execution speed.

For details on this subject, refer to Chapter 5, "Instruction Set."

1.3.4 Faster Context Switching

The Z380 CPU architecture allows multiple sets of register banks for AF/AF', BC/DE/HL, BC'/DE'/HL', IX/IX', IY/IY' register pairs (including each register's Extended portion). When doing context switching, by exceptional condition (trap or interrupts) or by subroutine/procedure calls, the CPU has to save the contents of the registers currently in use, along with the current CPU status.

Traditionally in the Z80 CPU architecture, this is done by saving the contents of the register into memory, usually using push/pop instructions or the auxiliary register file. Register contents are then restored when the process is finished.

With the Z380 CPU's multiple register banks, saving the contents of the working register set currently in use is just a matter of an instruction to change the field in the Select Register, which allows fast context switching.

1.4 SUMMARY

The Z380 CPU is a high-performance 16-bit Central Processing Unit Superintegration™ core. Code-compatible with the Z80 CPU, the Z380 CPU architecture has been expanded to include features such as multiple register banks, 4 Gbytes of linear memory addressing space, and efficient handling of nested interrupts. The benefits of this architecture, including high throughput rates, code density, and compiler efficiency, greatly enhance the power and versatility of the Z380 CPU. Thus, the Z380 CPU provides both a growth path for existing Z80-based designs and a powerful processor for applications and the products to be developed around this CPU core.

Table of Contents

Z380™ Architectural Overview 1

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

CHAPTER 2

ADDRESS SPACES

2.1 INTRODUCTION

The Z380 CPU supports five address spaces corresponding to the different types of locations that can be addressed and the method by which the logical addresses are formed. These five address spaces are:

- CPU Register Space. This consists of all the register addresses in the CPU register file.
- CPU Control Register Space. This consists of the Select Register (SR).
- Memory Address Space. This consists of the addresses of all locations in the main memory.
- External I/O Address Space. This consists of all external I/O ports addresses through which peripheral devices are accessed.
- On-Chip I/O Address Space. This consists of all internal I/O port addresses through which peripheral devices are accessed. Also, this addressing space contains registers to control the functionality of the device, giving status information.

2.2 CPU REGISTER SPACE

The Z380 register file is illustrated in Figure 2-1. Note that this figure shows the configuration of the register on the Z380 CPU, and the number of the register files may vary on future Superintegration devices. The Z380 CPU contains abundant register resources. At any given time, the program has immediate access to both primary and alternate registers in the selected register set. Changing register sets is a simple matter of an LDCTL instruction to program the Select Register (SR).

The CPU register file is divided into five groups of registers (an apostrophe indicates a register in the auxiliary registers).

- Four sets of Flag and Accumulator registers (F, A, F', A')
- Four sets of Primary and Working registers (B, C, D, E, H, L, B', C', D', E', H', L')
- Four sets of Index registers (IX, IY, IX', IY')
- Stack Pointer (SP)
- Program Counter, Interrupt register, Refresh register (PC, I, R)

Register addresses are either specified explicitly in the instruction or are implied by the semantics of the instruction.

2.2 CPU REGISTER SPACE (Continued)

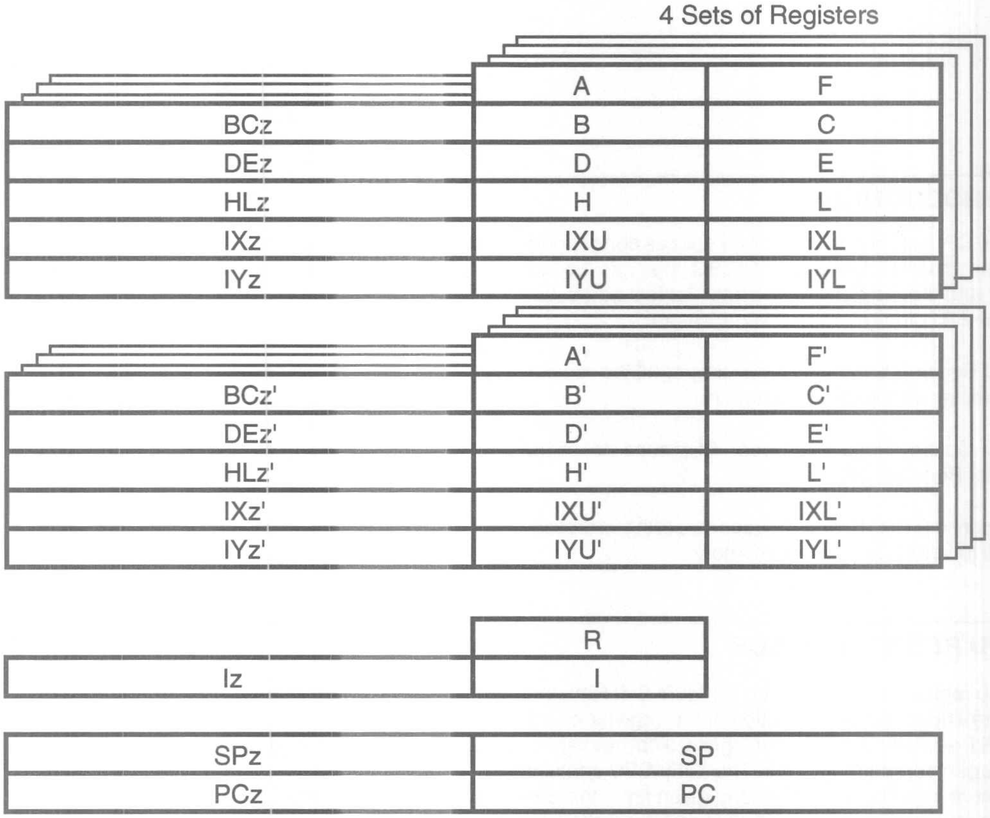

4 Sets of Registers
A
F
BCz
B
C
DEz
D
E
HLz
H
L
IXz
IXU
IXL
IYz
IYU
IYL
A'
F'
BCz'
B'
C'
DEz'
D'
E'
HLz'
H'
L'
IXz'
IXU'
IXL'
IYz'
IYU'
IYL'
R
Iz
I
SPz
SP
PCz
PC

Figure 2-1. Register File Organization (Z380 MPU)

2.2.1 Primary and Working Registers

The working register set is divided into two register files: the primary file and the alternate file (designated by prime ('')). Each file contains an 8-bit accumulator (A), a Flag register (F), and six 8-bit general-purpose registers (B, C, D, E, H, and L) with their Extended registers. Only one file can be active at any given time, although data in the inactive file can still be accessed by using EX R, R' instructions for the byte-wide registers, EX RR, RR' instructions for register pairs (either in 16-bit or 32-bit wide depending on the LW status). Exchange instructions allow the programmer to exchange the active file with the inactive file. The EX AF, AF', EXX, or EXALL instructions changes the register files in use. Upon reset, the primary register file in register set 0 is active. Changing register sets is a simple matter of an LDCTL instruction to program SR.

The accumulator is the destination register for 8-bit arithmetic and logical operations. The six general-purpose registers can be paired (BC, DE, and HL), and are extended to 32 bits by the extension to the register (with suffix "z"; BCz/DEz/HLz), to form three 32-bit general-purpose registers. The HL register serves as the 16-bit or 32-bit accumulator for word operations. Access to the Extended portion of the registers is possible using the SWAP instruction or word Load instructions in Long Word operation mode.

The Flag register contains eight status flags. Four can be individually used for control of program branching, two are used to support decimal arithmetic, and two are reserved. These flags are set or reset by various CPU operations. For details on Flag operations, refer to Section 5.2, "Flag Register."

2.2.2. Index Registers

The four index registers, IX, IX', IY, and IY', are extended to 32 bits by the extension to the register (with suffix "z"; IXz/IYz), to form 32-bit index registers. To access the Extended portion of the registers use the SWAP instruction or word Load instructions in Long Word operation mode. These Index registers hold a 32-bit base address that is used in the Index addressing mode.

Only one register of each can be active at any given time, although data in the inactive file can still be accessed by using EX IX, IX' and EX IY, IY' (either in 16-bit or 32-bit wide depending on the LW bit status). Index registers can also function as general-purpose registers with the upper and lower bytes of the lower 16 bits being accessed individually. These byte registers are called IXU, IXU', IXL, and IXL' for the IX and IX' registers, and IYU, IYU', IYL, and IYL' for the IY and IY' registers.

Selection of primary or auxiliary Index registers can be made by EXXX, EXXY, or EXALL instructions, or programming of SR. Upon reset, the primary registers in register set 0 is active. Changing register sets is a simple matter of an LDCTL instruction to program SR.

2.2.3. Interrupt Register

The Interrupt register (I) is used in interrupt modes 2 and 3 for /INT0 to generate a 32-bit indirect address to an interrupt service routine. The I register supplies the upper 24 or 16 bits of the indirect address and the interrupting peripheral supplies the lower eight or 16 bits. In Assigned Vectors mode for /INT3-/INT1, the upper 16 bits of the vector are supplied by the I register; bits 15-9 are supplied from the Assigned Vector Base register, and bits 8-0 are the assigned vector unique to each of /INT3-/INT1.

2.2.4. Program Counter

The Program Counter (PC) is used to sequence through instructions in the currently executing program and to generate relative addresses. The PC contains the 32-bit address of the current instruction being fetched from memory. In Native mode, the PC is effectively only 16 bits long, since the upper word [PC31-PC16] of the PC is forced to zero, and when carried from bit 15 to bit 16 (Lower word [PC15-PC0] to Upper word [PC31-PC16]) are inhibited in this mode. In Extended mode, the PC is allowed to increment across all 32 bits.

2.2.5. R Register

The R register can be used as a general-purpose 8-bit read/write register. The R register is not associated with the refresh controller and its contents are changed only by the user.

2.2.6. Stack Pointer

The Stack Pointer (SP) is used for saving information when an interrupt or trap occurs and for supporting subroutine calls and returns. Stack Pointer relative addressing allows parameter passing using the SP. The SP is 16 bits wide, but is extended by the SPz register to 32 bits wide.

2.2.6 Stack Pointer (Continued)

Increment/decrement of the Stack Pointer is affected by modes of operation (Native or Extended). In Native mode, the stack operates in modulo 216, and in Extended mode, it operates in modulo 232. For example, SP holds 0001FFFEH, and does the Word size Pop operation. After the operation, SP holds 00010000H in Native mode, and 00020000H in Extended mode. In either case, SPz can be programmed to set Stack frame. This is done by the Load-to-Stack pointer instructions in Long Word mode.

2.3. CPU CONTROL REGISTER SPACE

The CPU control register space consists of the 32-bit Select Register (SR). The SR may be accessed as a whole or the upper three bytes of the SR may be accessed individually as YSR, XSR, and DSR. In addition, these upper three bytes can be loaded with the same byte value. The SR may also be PUSHed and POPed and is cleared to zeros on Reset. For details on this register, refer to Chapter 5.3, "Select Register."

2.4 MEMORY ADDRESS SPACE

The memory address space can be viewed as a string of 4 Gbytes numbered consecutively in ascending order. The 8-bit byte is the basic addressable element in the Z380 MPU memory address space. However, there are other addressable data elements: bits, 2-byte words, byte strings, and 4-byte words.

The size of the data element being addressed depends on the instruction being executed as well as the Word/Long Word mode. A bit can be addressed by specifying a byte and a bit within that byte. Bits are numbered from right to left, with the least significant bit being 0, as illustrated in Figure 2-2.

The address of a multiple-byte entity is the same as the address of the byte with the lowest memory address in the entity. Multiple-byte entities can be stored beginning with either even or odd memory addresses. A word (either 2-byte or 4-byte entity) is aligned if its address is even; otherwise it is unaligned. Multiple bus transactions, which may be required to access multiple-byte entities, can be minimized if alignment is maintained.

The format of multiple-byte data types is also shown in Figure 2-2. Note that when a word is stored in memory, the least significant byte precedes the more significant byte of the word, as in the Z80 CPU architecture. Also, the lower-addressed byte is present on the upper byte of the external data bus.

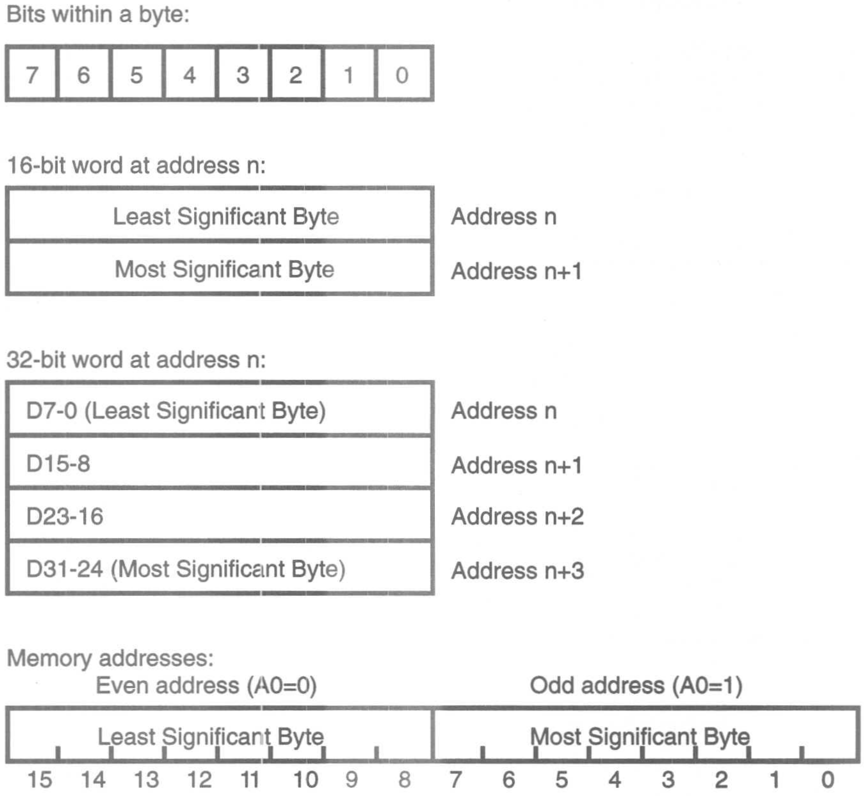

Bits within a byte:
7
6
5
4
3
2
1
0
16-bit word at address n:
Least Significant Byte
Address n
Most Significant Byte
Address n+1
32-bit word at address n:
D7-0 (Least Significant Byte)
Address n
D15-8
Address n+1
D23-16
Address n+2
D31-24 (Most Significant Byte)
Address n+3
Memory addresses:
Even address (A0=0)
Odd address (A0=1)
Least Significant Byte
Most Significant Byte
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Figure 2-2. Bit/Byte Ordering Conventions

2.5. EXTERNAL I/O ADDRESS SPACE

External I/O address space is 4 Gbytes in size and External I/O addresses are generated by I/O instructions except those reserved for on-chip I/O address space accesses. It can take a variety of forms, as shown in Table 2.1. An external I/O read or write is always one transaction, regardless of the bus size and the type of I/O instruction.

Table 2-1. I/O Addressing Options

	Address Bus			
I/O Instruction	**A31-A24**	**A23-A16**	**A15-A8**	**A7-A0**
IN A, (n)	00000000	00000000	A7-A0	n
IN dst,(C)	BC31-B24	BC23-B16	BC15-B8	BC7-B0
INA(W) dst,(mn)	00000000	00000000	m	n
DDIR IB INA(W) dst,(lmn)	00000000	l	m	n
DDIR IW INA(W) dst,(klmn)	k	l	m	n
Block Input	BC31-B24	BC23-B16	BC15-B8	BC7-B0
OUT (n),A	00000000	00000000	A7-A0	n
OUT (C),dst	BC31-B24	BC23-B16	BC15-B8	BC7-B0
OUTA(W) (mn),dst	00000000	00000000	m	n
DDIR IB OUTA(W) (lmn),dst	00000000	l	m	n
DDIR IW OUTA(W) (klmn),dst	k	l	m	n
Block Output	BC31-B24	BC23-B16	BC15-B8	BC7-B0

2.6. ON-CHIP I/O ADDRESS SPACE

The Z380 CPU has the on-chip I/O address space to control on-chip peripheral functions of the Superintegration™ version of the devices. A portion of its interrupt functions are also controlled by several on-chip registers, which occupy an on-chip I/O address space. This on-chip I/O address space can be accessed only with the following reserved on-chip I/O instructions which are identical to the Z180 original I/O instructions to access Page 0 I/O addressing area.

IN0 R,(n)
IN0 (n)
OUT0 (n),R
TSTIO n
OTIM
OTIMR
OTDM
OTDMR

When one of these I/O instructions is executed, the Z380 MPU outputs the register address being accessed in a pseudo-transaction of two BUSCLK cycles duration, with the address signals A31-A8 at zero. In the pseudo-transactions, all bus control signals are at their inactive state.

The following four registers are assigned to this addressing space as a part of the Z380 CPU core:

Register Name	**Internal I/O Address**
Interrupt Enable Register	17H
Assigned Vector Base Register	18H
Trap and Break Register	19H
Chip Version ID Register	0FFH

The Chip Version ID register returns one byte data, which indicates the version of the CPU, or the specific implementation of the Z380 CPU based Superintegration device. Currently, the value 00H is assigned to the Z380 MPU, and other values are reserved.

For the other three registers, refer to Chapter 6, "Interrupts and Traps."

Also, the Z380 MPU has registers to control chip selects, refresh, waits, and I/O clock divide to Internal I/O address 00H to 10H. For these registers, refer to the Z380 MPU Product specification (DC-3003-01).

Table of Contents

Z380™ Architectural Overview 1

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

CHAPTER 3

NATIVE EXTENDED MODE, WORD/LONG WORD MODE OF OPERATIONS AND DECODER DIRECTIONS

3.1 INTRODUCTION

The Z380™ CPU architecture allows access to 4 Gbytes (2^{32}) of memory addressing space, and 4G locations of I/O. It offers 16/32-bit manipulation capability while maintaining object-code compatibility with the Z80 CPU. In order to implement these capabilities and new instruction sets, it has two modes of operation for address manipulation (Native or Extended mode), two modes of operation for data manipulation (Word or Long Word mode), and a special set of new Decoder Directives.

On Reset, the Z380 CPU defaults in Native mode and Word mode. In this condition, it behaves exactly the same as the Z80 CPU, even though it has access to the entire 4 Gbytes of memory for data access and 4G locations of I/O space, access to the newly added registers which includes Extended registers and register banks, and the capability of executing all the Z380 instructions.

As described below, the Z380 CPU can be switched between Word mode and Long Word mode during operation through the SETC LW and RESC LW instructions, or Decoder Directives. The Native and Extended modes are a key exception— it defaults up in Native mode, and can be set to Extended mode by the instruction. Only Reset can return it to Native mode. Figure 3-1 illustrates the relationship between these modes of operation.

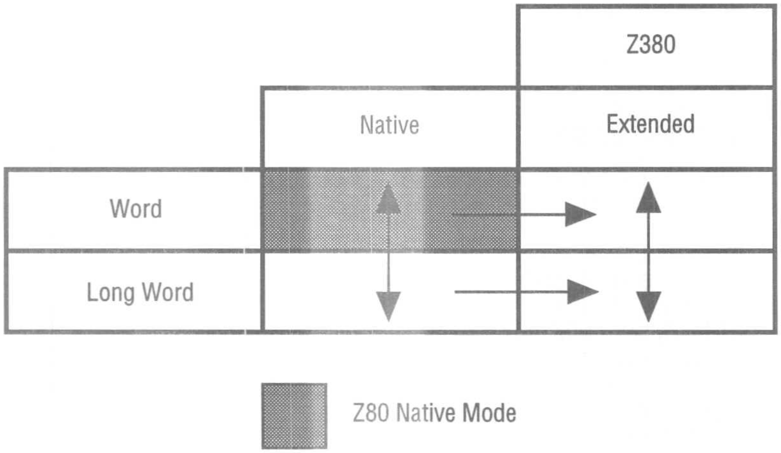


Figure 3-1. Z380™ CPU Operation Modes

For the instructions which work with the DDIR instructions, refer to Appendix D and E.

3.2 DECODER DIRECTIVES

The Decoder Directive is not an instruction, but rather a directive to the instruction decoder. The instruction decoder may be directed to fetch an additional byte or word of immediate data or address with the instruction, as well as tagging the instruction for execution in either Word or Long Word mode. Since the Z80 CPU architecture's addressing convention in the memory is "least significant byte first, followed by more significant bytes," it is possible to have such instructions to direct the instruction decoder to fetch additional byte(s) of address information or immediate data to extend the instruction.

All eight combinations of the two options are supported, as shown below. Instructions which do not support decoder directives are assembled by the instruction decoder as if the decoder directive were not present.

■	DDIR W	Word mode
■	DDIR IB,W	Immediate byte, Word mode
■	DDIR IW,W	Immediate Word, Word mode
■	DDIR IB	Immediate byte
■	DDIR LW	Long Word mode
■	DDIR IB,LW	Immediate byte, Long Word mode
■	DDIR IW,LW	Immediate Word, Long Word mode
■	DDIR IW	Immediate Word

The IB decoder directive causes the decoder to fetch an additional byte immediately after the existing immediate data or direct address, and in front of any trailing opcode bytes (with instructions starting with DD-CB or FD-CB, for example).

Likewise, the IW decoder directive causes the decoder to fetch an additional word immediately after the existing immediate data or direct address, and in front of any trailing opcode bytes.

Byte ordering within the instruction follows the usual convention; least significant byte first, followed by more significant bytes. More-significant immediate data or direct address bytes not specified in the instruction are read as all zeros by the processor.

The W decoder directive causes the instruction decoder to tag the instruction for execution in Word mode. This is useful while the Long Word (LW) bit in the Select Register (SR) is set, but 16-bit data manipulation is required for this instruction.

The LW decoder directive causes the instruction decoder to tag the instruction for execution in Long Word mode. This is useful while the LW bit in the SR is cleared, but 32-bit data manipulation is required for this instruction.

3.3 NATIVE MODE AND EXTENDED MODE

The Z380 CPU can operate in either Native or Extended mode, as a way to manipulate addresses.

In Native mode (the Reset configuration), the Program Counter only increments across 16 bits, and all stack Push and Pop operations manipulate 16-bit quantities (two bytes). Thus, Native mode is fully compatible with the Z80 CPU's 64 Kbyte address space and programming model. The extended portion of the Program Counter (PC31-PC15) is forced to 0 and program address location next to 0000FFFFH is 00000000H in this mode. This means in Native mode, program have to reside within the first 64 Kbytes of the memory addressing space.

In Extended mode, however, the PC increments across all 32 bits and all stack Push and Pop operations manipulate 32-bit quantities. Thus, Extended mode allows access to the entire 4 Gbyte address space. In both Native and Extended modes, the Z380 CPU drives all 32 bits of the address onto the external address bus; only the PC increments and stack operations distinguish Native from Extended mode.

Note that regardless of Native or Extended mode, a 32-bit address is always used for the data access. Thus, for data reference, the complete 4 Gbytes of memory area may be accessed. For example:

```
        LD      BC, (HL)
```

uses the 32-bit address value stored in HL31-HL0 (HLz and HL) as a source location address. However, on Reset, the HL31-HL16 portion (HLz) initializes to 00H. Unless HLz is modified to other than 00H, operation of this instruction is identical to the one with the Z80 CPU. Modifying the extended portion of the register is done either by using a 32-bit load instruction (in Long Word mode, or with DDIR LW instructions), or using a 16-bit load instruction with SWAP instructions.

The Z380 CPU implements one instruction to switch to Extended mode from Native mode; SETC XM (set Extended mode) places the Z380 CPU in Extended mode. Once in Extended mode, only Reset can return it to Native mode. On Reset, the Z380 is in Native mode. Refer to Sections 4 and 5 for more examples.

3.4 WORD AND LONG WORD MODE OF OPERATION

The Z380 CPU can operate in either Word or Long Word mode. In Word mode (the Reset configuration), all word operations manipulate 16-bit quantities, and are compatible with the Z80 CPU 16-bit operations. In the Long Word mode, all word operations can manipulate 32-bit quantities. Note that the Native/Extended and Word/Long Word selections are independent of one another, as Word/Long Word pertains to data and operand address manipulation only. The Z380 CPU implements two instructions and two decoder directives to allow switching between these two modes; SETC LW (Set Long Word) and RESC LW (Reset Long Word) perform a global switch, while DDIR LW and DDIR W are decoder directives that select a particular mode only for the instruction that they precede.

Examples:

1. Effect of Word mode and Long Word mode

```
DDIR W
LD      BC, (HL)
```

   Loads BC15-BC0 from the location (HL) and (HL+1), and BCz (BC31-BC16) remains unchanged.

```
DDIR LW
LD      BC, (HL)
```

   Loads BC31-BC0 from the locations (HL) to (HL+3).

2. Immediate data load with DDIR instructions

```
DDIR IW,LW
LD      HL,12345678H
```

   Loads 12345678H into HL31-HL0.

```
DDIR IB,LW
LD      HL,123456H
```

   Loads 00123456H into HL31-HL0.
   00H is appended as the Most significant byte as HL31-HL24.

```
DDIR LW
LD      HL,1234H
```

   Loads 00001234H into HL31-HL0.
   0000H is appended as the HL31-HL16 portion.

ZILOG

Table of Contents

Z380™ Architectural Overview 1

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

CHAPTER 4
ADDRESSING MODES AND DATA TYPES

4.1 INSTRUCTION

An instruction is a consecutive list of one or more bytes in memory. Most instructions act upon some data; the term operand refers to the data to be operated upon. For Z380™ CPU instructions, operands can reside in CPU registers, memory locations, or I/O ports (internal or external). The method used to designate the location of the operands for an instruction are called addressing modes. The Z380 CPU supports seven addressing modes; Register, Immediate, Indirect Register, Direct Address, Indexed, Program Counter Relative Address, and Stack Pointer Relative. A wide variety of data types can be accessed using these addressing modes.

4.2 ADDRESSING MODE DESCRIPTIONS

The following pages contain descriptions of the addressing modes for the Z380 CPU. Each description explains how the operand's location is calculated, indicates which address spaces can be accessed with that particular addressing mode, and gives an example of an instruction using that mode, illustrating the assembly language format for the addressing modes.

4.2.1 Register (R, RX)

When this addressing mode is used, the instruction processes data taken from one of the 8-bit registers A, B, C, D, E, H, L, IXU, IXL, IYU, IYL, one of the 16-bit registers BC, DE, HL, IX, IY, SP, or one of the special byte registers I or R.

Storing data in a register allows shorter instructions and faster execution that occur with instructions that access memory.

Instruction
OPERATION REGISTER → OPERAND

The operand value is the contents of the register.

The operand is always in the register address space. The register length (byte or word) is specified by the instruction opcode. In the case of Long Word register operation, it is specified either through the SETC LW instruction or the DDIR LW decoder directive.

Example of R mode:

1. **Load register in Word mode.**

```
DDIR W     ;Next instruction in Word mode
LD BC,HL   ;Load the contents of HL into BC
```

	BCz	BC	HLz	HL
Before instruction execution	1234	5678	9ABC	DEF0
After instruction execution	1234	DEF0	9ABC	DEF0

2. **Load register in Long Word mode.**

```
DDIR LW    ;Next instruction in Long Word mode
LD BC,HL   ;Load the contents of HL into BC
```

	BCz	BC	HLz	HL
Before instruction execution	1234	5678	9ABC	DEF0
After instruction execution	9ABC	DEF0	9ABC	DEF0

4.2.2 Immediate (IM)

When the Immediate addressing mode is used, the data processed is in the instruction.

The Immediate addressing mode is the only mode that does not indicate a register or memory address as the source operand.

4.2.2 Immediate (IM) (Continued)

Instruction
OPERATION
OPERAND

The operand value is in the instruction

Immediate mode is often used to initialize registers. Also, this addressing mode is affected by the DDIR Immediate Data Directives to expand the immediate value to 24 bits or 32 bits.

Example of IM mode:

1. **Load immediate value into accumulator**
   LD A,55H ;Load hex 55 into the accumulator.

	A
Before instruction execution	12
After instruction execution	55

2. **Load 24-bit immediate value into HL register**

DDIR IB, LW	;next instruction is in Long Word mode, with ;an additional immediate data
LD HL, 123456H	;load HLz, and HL with constant 123456H

This case, the Z380 CPU appends 00H as a MSB byte.

	HLz	HL
Before instruction execution	0987	6543
After instruction execution	0012	3456

4.2.3 Indirect Register (IR)

In Indirect Register addressing mode, the register specified in the instruction holds the address of the operand. The data to be processed is in the location specified by the BC, DE, or HL register (depending on the instruction) for memory accesses, or C register for I/O.

Memory or
Instruction
OPERATION REGISTER → Register Address → I/O Port OPERAND

The operand value is the contents of the location whose address is in the register.

Depending on the instruction, the operand specified by IR mode is located in either the I/O address space (I/O instruction) or memory address space (all other instructions).

Indirect Register mode can save space and reduce execution time when consecutive locations are referenced or one location is repeatedly accessed. This mode can also be used to simulate more complex addressing modes, since addresses can be computed before data is accessed.

The address in this mode is always treated as a 32-bit mode. After reset, the contents of the extend registers (registers with "z" suffix) are initialized as 0's; hence, these instructions will be executed just as for the Z80/Z180.

Example of IR mode:

1. **Load accumulator from the contents of memory pointed by (HL)**
   LD A, (HL) ;Load the accumulator with the data
   ;addressed by the contents of HL

	A	HLz,HL
Before instruction execution	0F	12345678
After instruction execution	0B	12345678

Memory location	12345678	0B

4.2.4 Direct Address (DA)

When Direct Address mode is used, the data processed is at the location whose memory or I/O port address is in the instruction.

Instruction		Memory or
OPERATION		I/O Port
ADDRESS	→	OPERAND

The operand value is the contents of the location whose address is in the instruction.

Depending on the instruction, the operand specified by DA mode is either in the I/O address space (I/O instruction) or memory address space (all other instructions).

This mode is also used by Jump and Call instructions to specify the address of the next instruction to be executed. (The address serves as an immediate value that is loaded into the program counter.)

Also, DDIR Immediate Data Directives are used to expand the direct address to 24 or 32 bits. Operand width is affected by LW bit status for the load and exchange instructions.

Example of DA mode:

1. **Load BC register from memory location 00005E22H in Word mode**

```
LD  BC, (5E22H)     ;Load BC with the data in address
                    ;00005E22H
```

	BC
Before instruction execution	1234
After instruction execution	0301

Memory location	00005E22	01
	00005E23	03

2. **Load BC register from memory location 12345E22H in Word mode**

```
DDIR IW                 ;extend direct address by one word
LD  BC, (12345E22H)     ;Load BC with the data in address
                        ;12345E22H
```

	BC
Before instruction execution	1234
After instruction execution	0301

Memory location	12345E22	01
	12345E23	03

3. **Load BC register from memory location 12345E22H in Long Word mode**

```
DDIR IW,LW              ;extend direct address by one word,
                        ;and operation in Long Word
LD  BC, (12345E22H)     ;Load BC with the data in address
                        ;12345E22H
```

	BCz	BC
Before instruction execution	1234	5678
After instruction execution	0705	0301

Memory location	12345E22	01
	12345E23	03
	12345E24	05
	12345E25	07

4.2.5 Indexed (X)

When the Indexed addressing mode is used, the data processed is at the location whose address is the contents of IX or IY in use, offset by an 8-bit signed displacement in the instruction.

The Indexed address is computed by adding the 8-bit two's complement signed displacement specified in the instruction to the contents of the IX or IY register in use, also specified by the instruction. Indexed addressing allows random access to tables or other complex data structures where the address of the base of the table is known, but the particular element index must be computed by the program.

The offset portion can be expanded to 16 or 24 bits, instead of eight bits by using DDIR Immediate Data Directives (DDIR IB for 16-bit offset, DDIR IW for 24-bit offset).

Note that computation of the effective address is affected by the operation mode (Native or Extended). In Native mode, address computation is done in modulo 2^{16}, and in Extended mode, address computation is done in modulo 2^{32}.

```
Instruction                                           REGISTER    MEMORY
OPERATION REGISTER →      ADDRESS          →+         OPERAND
DISPLACEMENT          ________________________________            ↑
```

Example of X mode:

1. Load accumulator from location (IX-1) in Native mode

```
LD A, (IX-1)        ;Load into the accumulator the
                    ;contents of the memory location
                    ;whose address is one less than
                    ;the contents of IX
                    ;Assume it is in Native mode
```

	A	IXz	IX
Before instruction execution	01	0001	0000
After instruction execution	23	0001	0000
Memory location	0001FFFF	23	

Address calculation: In Native mode, 0FFH encoding in the instruction is sign extended to a 16-bit value before the address calculation, but calculation is done in modulo 2^{16} and does not take into account the index register's extended portion.

$$\begin{array}{r} 0000 \\ +\ \underline{FFFF} \\ FFFF \end{array}$$

2. Load accumulator from location (IX-1) in Extended mode

```
SETC   XM          ;Set Extended mode
LD     A, (IX-1)   ;Load into the accumulator the
                   ;contents of the memory location
                   ;whose address is one less than
                   ;the contents of IX
```

	A	IXz	IX
Before instruction execution	01	0001	0000
After instruction execution	23	0001	0000
Memory location	0000FFFF	23	

Address calculation: In Extended mode, 0FFH encoding in the instruction is sign extended to a 32-bit value before the address calculation, but calculation is done in modulo 2^{32} and takes into account the index register's extended portion.

```
   00010000
+  FFFFFFFF
   --------
   0000FFFF
```

4.2.6 Program Counter Relative Mode (RA)

The Program Counter Relative Addressing mode is used by certain program control instructions to specify the address of the next instruction to be executed (specifically, the sum of the Program Counter value and the displacement value is loaded into the Program Counter). Relative addressing allows reference forward or backward from the current Program Counter value; it is used for program control instructions such as Jumps and Calls that access constants in the memory.

As a displacement, an 8-bit, 16-bit, or 24-bit value can be used. The address to be loaded into the Program Counter is computed by adding the two's complement signed displacement specified in the instruction to the current Program Counter.

Also, in Native mode,

```
Instruction     PC                 MEMORY
OPERATION       ADDRESS     -→+    OPERAND
DISPLACEMENT                --↑
```

Note that computation of the effective address is affected by the mode of operation (Native or Extended). In Native mode, address computation is done in modulo 2^{16}, and the PC Extend (PC31-PC16) is forced to 0 and will not affect this portion. In Extended mode, address computation is done is modulo 2^{32}, and will affect the contents of PC extend if there is a carry or borrow operation.

Example of RA mode:

1. Jump relative in Native mode, 8-bit displacement

```
JR     $-2     ;Jumps to the location
               ;(Current PC value) – 2
               ;'$' represents for current PC value
               ;This instruction jumps to itself.
               ;since after the execution of this instruction,
               ;PC points to the next instruction.
```

4.2.6 Program Counter Relative Mode (RA) (Continued)

	PCz	PC
Before instruction execution	0000	1000
After instruction execution	0000	0FFE

Address calculation: In Native mode, –2 is encoded as 0FEH in the instruction, and it is sign extended to a 16-bit value before added to the Program Counter. Calculation is done in modulo 2^{16} and does not affect the Extended portion of the Program Counter.

```
    1000
+   FFFE
    FFFE
```

2. Jump relative in Extended mode, 16-bit displacement

```
SETC   XM            ;Put it in Extended mode of operation
JR     $-5000H       ;Jumps to the location
                     ;(Current PC value) – 5000H
                     ;$ stands for current PC value
                     ;This instruction jumps to itself.
```

	PCz	PC
Before instruction execution	1959	0807
After instruction execution	1958	B80B

Address calculation: Since this is a 4-byte instruction, the PC value after fetch but before jump taking place is:

```
    19590807
+   00000004
    1959080B
```

The displacement portion, –5000H, is sign extended to a 32-bit value before being added to the Program Counter. Calculation is done in modulo 2^{32} and affects the Extended portion of the Program Counter.

```
    1959080B
+   FFFFB000
    1958B80B
```

4.2.7 Stack Pointer Relative Mode (SR)

For Stack Pointer Relative addressing mode, the data processed is at the location whose address is the contents of the Stack Pointer, offset by an 8-bit displacement in the instruction.

The Stack Pointer Relative address is computed by adding the 8-bit two's complement signed displacement specified in the instruction to the contents of the SP, also specified by the instruction. Stack Pointer Relative addressing mode is used to specify data items to be found in the stack, such as parameters passed to procedures.

Offset portion can be expanded to 16 or 24 bits by using DDIR immediate instructions (DDIR IB for a 16-bit offset, DDIR IW for a 24-bit offset).

Note that computation of the effective address is affected by the operation mode (Native or Extended). In Native mode, address computation is done in modulo 2^{16}, meaning computation is done in 16-bit and does not affect upper half of the SP portion for calculation (wrap around within the 16-bit). In Extended mode, address computation is done in modulo 2^{32}.

Also, the size of the data transfer is affected by the LW mode bit. In Word mode, transfer is done in 16 bits, and in Long Word mode, transfer is done in 32 bits.

Instruction
OPERATION
DISPLACEMENT

SP
ADDRESS —I
—+

MEMORY
OPERAND

Example of SR mode:

1. **Load HL from location (SP – 4) in Native mode, Word mode**

LD HL, (SP–4) ;Load into the HL from the
;contents of the memory location
;whose address is four less than
;the contents of SP.
;Assume it is in Native/Word mode.

	HLz	HL	SPz	SP
Before instruction execution	1234	5678	07FF	7F00
After instruction execution	EFCD	AB89	07FF	7F00

Memory location	07FF7EFC	89
	07FF7EFD	AB

Address calculation: In Native mode, FCH (–4 in Decimal) encoding in the instruction is sign extended to a 16-bit value before the address calculation. Calculation is done in modulo 2^{16} and does not take into account the Stack Pointer's extended portion.

$$\begin{array}{rr} & 7F00 \\ + & \underline{FFFC} \\ & 7EFC \end{array}$$

4.2.7 Stack Pointer Relative Mode (SR) (Continued)

2. Load HL from location (SP – 4) in Extended mode, Long Word mode

```
SETC      XM              ;In Extended mode
DDIR LW                   ;operate next instruction in Long Word mode
LD  HL, (SP–4)            ;Load into the HL from the
                          ;contents of the memory location
                          ;whose address is four less than
                          ;the contents of SP.
```

	HLz	HL	SPz	SP
Before instruction execution	1234	5678	07FF	7F00
After instruction execution	EFCD	AB89	07FF	7F00

Memory location	07FF7EFC	89
	07FF7EFD	AB
	07FF7EFE	CD
	07FF7EFF	EF

Address calculation: In Extended mode, FCH (–4 in Decimal) encoding in the instruction is sign extended to a 32-bit value before the address calculation, and calculation is done in modulo 2^{32}.

```
   07FF7F00
+  FFFFFFFC
   07FF7EFC
```

3. Load HL from location (SP + 10000H) in Extended mode, Long Word mode

```
SETC      XM              ;In Extended mode,
DDIR      IW,LW           ;operate next instruction in Long Word mode
                          ;with a word immediate data.
LD HL, (SP+10000)         ;Load into the HL from the
                          ;contents of the memory location
                          ;whose address is 10000H more than
                          ;the contents of SP.
```

	HLz	HL	SPz	SP
Before instruction execution	1234	5678	07FF	7F00
After instruction execution	EFCD	AB89	07FF	7F00

Memory location	08007F00	89
	08007F01	AB
	08007F02	CD
	08007F03	EF

Address calculation: In Extended mode, 010000H encoding in the instruction is sign extended to a 32-bit value before the address calculation, and calculation is done in modulo 2^{32}.

```
   07FF7F00
+  00010000
   08007F00
```

4.3 DATA TYPES

The Z380 CPU can operate on bits, binary-coded decimal (BCD) digits (four bits), bytes (eight bits), words (16 bits or 32 bits), byte strings, and word strings. Bits in registers can be set, cleared, and tested.

The basic data type is a byte, which is also the basic accessible element in the register, memory, and I/O address space. The 8-bit load, arithmetic, logical, shift, and rotate instructions operate on bytes in registers or memory. Bytes can be treated as logical, signed numeric, or unsigned numeric value.

Words are operated on in a similar manner by the word load, arithmetic, logical, and shift and rotate instructions.

Operation on 2-byte words is also supported. Sixteen-bit load and arithmetic instructions operate on words in registers or memory; words can be treated as signed or unsigned numeric values. I/O reads and writes can be 8-bit or 16-bit operations. Also, the Z380 CPU architecture supports operation in Long Word mode to handle a 32-bit address manipulation. For that purpose, 16-bit wide registers originally on the Z80 have been expanded to 32 bits wide, along with the support of the arithmetic instruction needed for a 32-bit address manipulation.

Bits are fully supported and addressed by number within a byte (see Figure 2-2). Bits within byte registers or memory locations can be tested, set, or cleared.

Operation on binary-coded decimal (BCD) digits are supported by Decimal Adjust Accumulator (DAA) and Rotate Digit (RLD and RRD) instructions. BCD digits are stored in byte registers or memory locations, two per byte. The DAA instruction is used after a binary addition or subtraction of BCD numbers. Rotate Digit instructions are used to shift BCD digit strings in memory.

Strings of up to 65536 (64K) bytes of Byte data or Word data can be manipulated by the Z380 CPU's block move, block search, and block I/O instructions. The block move instructions allow strings of bytes/words in memory to be moved from one location to another. Block search instructions provide for scanning strings of bytes/words in memory to locate a particular value. Block I/O instructions allow strings of bytes or words to be transferred between memory and a peripheral device.

Arrays are supported by Indexed mode (with 8-bit, 16-bit, or 24-bit displacement). Stack is supported by the Indexed and the Stack Pointer Relative addressing modes, and by special instructions such as Call, Return, Push, and Pop.

ZiLOG

Table of Contents

Z380™ Architectural Overview 1

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

CHAPTER 5

INSTRUCTION SET

5.1 INTRODUCTION

The Z380™ CPU instruction set is a superset of the Z80 CPU and the Z180 MPU; the Z380 CPU is opcode compatible with the Z80 CPU/Z180 MPU. Thus, a Z80/Z180 program can be executed on a Z380 CPU without modification. The instruction set is divided into 12 groups by function:

- 8-Bit Load/Exchange Group
- 16/32-Bit Load, Exchange, SWAP and Push/Pop Group
- Block Transfers, and Search Group
- 8-Bit Arithmetic and Logic Operations
- 16/32-Bit Arithmetic Operations
- 8-Bit Bit Manipulation, Rotate and Shift Group
- 16-Bit Rotates and Shifts
- Program Control Group
- Input and Output Operations for External I/O Space
- Input and Output Operations for Internal I/O Space
- CPU Control Group
- Decoder Directives

This chapter describes the instruction set of the Z380 CPU. Flags and condition codes are discussed in relation to the instruction set. Then, the interpretability of instructions and trap are discussed. The last part of this chapter is a detailed description of each instruction, listed in alphabetical order by mnemonic. This section is intended as a reference for Z380 CPU programmers. The entry for each instruction contains a complete description of the instruction, including addressing modes, assembly language mnemonics, and instruction opcode formats.

5.2 PROCESSOR FLAGS

The Flag register contains six bits of status information that are set or cleared by CPU operations (Figure 5-1). Four of these bits are testable (C, P/V, Z, and S) for use with conditional jump, call, or return instructions. Two flags are not testable (H and N) and are used for binary-coded decimal (BCD) arithmetic.

S	Z	X	H	X	P/V	N	C
7	6	5	4	3	2	1	0

Figure 5-1. Flag Register

The Flag register provides a link between sequentially executed instructions, in that the result of executing one instruction may alter the flags, and the resulting value of the flags can be used to determine the operation of a subsequent instruction. The program control instructions, whose operation depends on the state of the flags, are the Jump, Jump Relative, subroutine Call, Call Relative, and subroutine Return instructions; these instructions are referred to as conditional instructions.

5.2.1 Carry Flag (C)

The Carry flag is set or cleared depending on the operation being performed. For add instructions that generate a carry and subtract instruction generating a borrow, the Carry flag is set to 1. The Carry flag is cleared to 0 by an add that does not generate a carry or a subtract that generates no borrow. This saved carry facilitates software routines for extended precision arithmetic. The multiply instructions use the Carry flag to signal information about the precision of the result. Also, the Decimal Adjust Accumulator (DAA) instruction leaves the Carry flag set to 1 if a carry occurs when adding BCD quantities.

For rotate instructions, the Carry flag is used as a link between the least significant and most significant bits for any register or memory location. During shift instructions, the Carry flag contains the last value shifted out of any register or memory location. For logical instructions the Carry flag is cleared. The Carry flag can also be set and complemented with explicit instructions.

5.2.2 Add/Subtract Flag (N)

The Add/Subtract flag is used for BCD arithmetic. Since the algorithm for correcting BCD operations is different for addition and subtraction, this flag is used to record when an add or subtract was last executed, allowing a subsequent Decimal Adjust Accumulator instruction to perform correctly. See the discussion of the DAA instruction for further information.

5.2.3 Parity/Overflow Flag (P/V)

This flag is set to a particular state depending on the operation being performed.

For signed arithmetic, this flag, when set to 1, indicates that the result of an operation on two's complement numbers has exceeded the largest number, or less than the smallest number, that can be represented using two's complement notation. This overflow condition can be determined by examining the sign bits of the operands and the result.

The P/V flag is also used with logical operations and rotate instructions to indicate the parity of the result. The of bits set to 1 in a byte are counted. If the total is odd, this flag is reset indicates odd parity (P = 0). If the total is even, this flag is set indicates even parity (P = 1).

During block search and block transfer instructions, the P/V flag monitors the state of the Byte Count register (BC). When decrementing the byte counter results in a zero value, the flag is cleared to 0; otherwise the flag is set to 1.

During Load Accumulator with I or R register instruction, the P/V flag is loaded with the IEF2 flag. For details on this topic,.refer to Chapter 6, "Interrupts and Traps."

When a byte is inputted to a register from an I/O device addressed by the C register, the flag is adjusted to indicate the parity of the data.

5.2.4 Half-Carry Flag (H)

The Half-Carry flag (H) is set to 1 or cleared to 0 depending on the carry and borrow status between bits 3 and 4 of an 8-bit arithmetic operation and between bits 11 and 12 of a 16-bit arithmetic operation. This flag is used by the Decimal Adjust Accumulator instruction to correct the result of an addition or subtraction operation on packed BCD data.

5.2.5 Zero Flag (Z)

The Zero flag (Z) is set to 1 if the result generated by the execution of certain instruction is a zero.

For arithmetic and logical operations, the Zero flag is set to 1 if the result is zero. If the result is not zero, the Zero flag is cleared to 0.

For block search instructions, the Zero flag is set to 1 if a comparison is found between the value in the Accumulator and the memory location pointed to by the contents of the register pair HL.

When testing a bit in a register or memory location, the Zero flag contains the complemented state of the tested bit (i.e., the Zero flag is set to 1 if the tested bit is a 0, and vice-versa).

For block I/O instructions, if the result of decrements B is zero, the Zero flag is set to 1; otherwise, it is cleared to 0. Also, for byte inputs to registers from I/O devices addressed by the C register, the Zero flag is set to 1 to indicate a zero byte input.

5.2.6 Sign Flag (S)

The Sign flag (S) stores the state of the most significant bit of the result. When the Z380 CPU performs arithmetic operation on signed numbers, binary two's complement notation is used to represent and process numeric information. A positive number is identified by a 0 in the most significant bit. A negative number is identified by a 1 in the most significant bit.

When inputting a byte from an I/O device addressed by the C register to a CPU register, the Sign flag indicates either positive (S = 0) or negative (S = 1) data.

5.2.7 Condition Codes

The Carry, Zero, Sign, and Parity/Overflow flags are used to control the operation of the conditional instructions. The operation of these instructions is a function of the state of one of the flags. Special mnemonics called condition codes are used to specify the flag setting to be tested during execution of a conditional instruction; the condition codes are encoded into a 3-bit field in the instruction opcode itself.

Table 5-1 lists the condition code mnemonic, the flag setting it represents, and the binary encoding for each condition code.

Table 5-1. Condition codes

Condition Codes for Jump, Call, and Return Instructions

Mnemonic	Meaning	Flag Setting	Binary Code
NZ	Not Zero*	Z = 0	000
Z	Zero*	Z = 1	001
NC	No Carry*	C = 0	010
C	Carry*	C = 1	011
NV	No Overflow	V = 0	100
PO	Parity Odd	V = 0	100
V	Overflow	V = 1	101
PE	Parity Even	V = 1	101
NS	No Sign	S = 0	110
P	Plus	S = 0	110
S	Sign	S = 1	111
M	Minus	S = 1	111

*Abbreviated set

Condition Codes for Jump Relative and Call Relative Instructions

Mnemonic	Meaning	Flag Setting	Binary Code
NZ	Not Zero	Z = 0	100
Z	Zero	Z = 1	101
NC	No Carry	C = 0	110
C	Carry	C = 1	111

5.3 SELECT REGISTER

The Select Register (SR) controls the register set selection and the operating modes of the Z380 CPU. The reserved bits in the SR are for future expansion; they will always read as zeros and should be written with zeros for future compatibility. Access to this register is done by using the newly added LDCTL instruction. Also, some of the instructions like EXX, IM p, and DI/EI change the bit(s). The SR was shown in Figure 5-2.

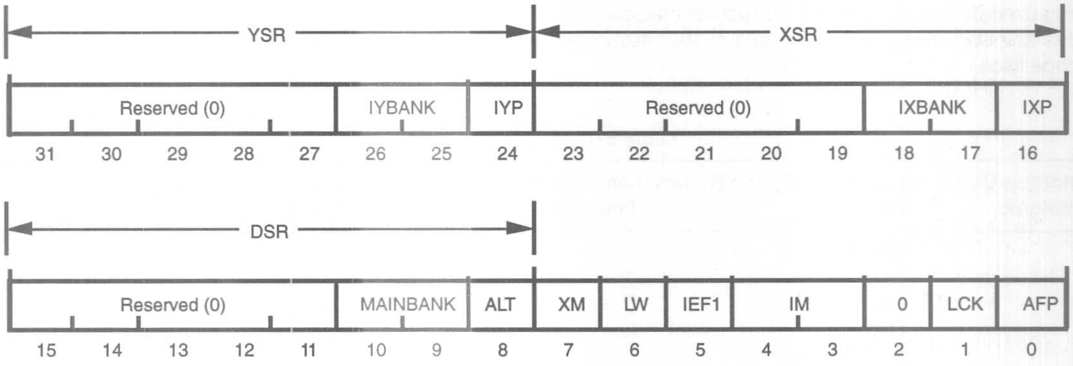


Figure 5-2. **Select Register**

5.3.1. IY Bank Select (IYBANK)

This 2-bit field selects the register set to be used for the IY and IY' registers. This field can be set independently of the register set selection for the other Z380 CPU registers. Reset selects Bank 0 for IY and IY'.

5.3.2. IY or IY' Register Select (IY')

This bit controls and reports whether IY or IY' is the currently active register. IY is selected when this bit is cleared, and IY' is selected when this bit is set. Reset clears this bit, selecting IY.

5.3.3. IX Bank Select (IXBANK)

This 2-bit field selects the register set to be used for the IX and IX' registers. This field can be set independently of the register set selection for the other Z380 CPU registers. Reset selects Bank 0 for IX and IX'.

5.3.4. IX or IX' Register Select (IX')

This bit controls and reports whether IX or IX' is the currently active register. IX is selected when this bit is cleared, and IX' is selected when this bit is set. Reset clears this bit, selecting IX.

5.3.5. Main Bank Select (MAINBANK)

This 2-bit field selects the register set to be used for the A, F, BC, DE, HL, A', F', BC', DE', and HL' registers. This field can be set independently of the register set selection for the other Z380 CPU registers. Reset selects Bank 0 for these registers.

5.3.6. BC/DE/HL or BC'/DE'/HL' Register Select (ALT)

This bit controls and reports whether BC/DE/HL or BC'/DE'/HL' is the currently active bank of registers. BC/DE/HL is selected when this bit is cleared, and BC'/DE'/HL' is selected when this bit is set. Reset clears this bit, selecting BC/DE/HL.

5.3.7. Extended Mode (XM)

This bit controls the Extended/Native mode selection for the Z380 CPU. This bit is set by the SETC XM instruction. This bit can not be reset by software, only by Reset. When this bit is set, the Z380 CPU is in Extended mode. Reset clears this bit, and the Z380 CPU is in Native mode.

5.3.8. Long Word Mode (LW)

This bit controls the Long Word/Word mode selection for the Z380 CPU. This bit is set by the SETC LW instruction and cleared by the RESC LW instruction. When this bit is set, the Z380 CPU is in Long Word mode; when this bit is cleared the Z380 CPU is in Word mode. Reset clears this bit. Note that individual Word load and exchange instructions may be executed in either Word or Long Word mode using the DDIR W and DDIR LW decoder directives.

5.3.9. Interrupt Enable Flag (IEF)

This bit is the master Interrupt Enable for the Z380 CPU. This bit is set by the EI instruction and cleared by the DI instruction, or on acknowledgment of an interrupt request. When this bit is set, interrupts are enabled; when this bit is cleared, interrupts are disabled. Reset clears this bit.

5.3.10. Interrupt Mode (IM)

This 2-bit field controls the interrupt mode for the /INT0 interrupt request. These bits are controlled by the IM instructions (00 = IM 0, 01 = IM 1, 10 = IM 2, 11 = IM 3). Reset clears both of these bits, selecting Interrupt Mode 0.

5.3.11. Lock (LCK)

This bit controls the Lock/Unlock status of the Z380 CPU. This bit is set by the SETC LCK instruction and cleared by the RESC LCK instruction. When this bit is set, no bus requests will be accepted, providing exclusive access to the bus by the Z380 CPU. When this bit is cleared, the Z380 CPU will grant bus requests in the normal fashion. Reset clears this bit.

5.3.12. AF or AF' Register Select (AF')

This bit controls and reports whether AF or AF' is the currently active pair of registers. AF is selected when this bit is cleared, and AF' is selected when this bit is set. Reset clears this bit, selecting AF.

5.4 INSTRUCTION EXECUTION AND EXCEPTIONS

Three types of exception conditions—interrupts, trap, and Reset—can alter the normal flow of program execution. Interrupts are asynchronous events generated by a device external to the CPU; peripheral devices use interrupts to request service from the CPU. Trap is a synchronous event generated internally in the CPU by executing undefined instructions. Reset is an asynchronous event generated by outside circuits. It terminates all current activities and puts the CPU into a known state. Interrupts and Traps are discussed in detail in Chapter 6, and Reset is discussed in detail in Chapter 7. This section examines the relationship between instructions and the exception conditions.

5.4.1 Instruction Execution and Interrupts

When the CPU receives an interrupt request, and it is enabled for interrupts of that class, the interrupt is normally processed at the end of the current instruction. However, the block transfer and search instructions are designed to be interruptible so as to minimize the length of time it takes the CPU to respond to an interrupt. If an interrupt request is received during a block move, block search, or block I/O instruction, the instruction is suspended after the current iteration. The address of the instruction itself, rather than the address of the following instruction, is saved on the stack, so that the same instruction is executed again when the interrupt handler executes an interrupt return instruction. The contents of the repetition counter and the registers that index into the block operands are such that, after each iteration, when the instruction is reissued upon returning from an interrupt, the effect is the same as if the instruction were not interrupted. This assumes, of course, that the interrupt handler preserves the registers.

5.4.2 Instruction Execution and Trap

The Z380 MPU generates a Trap when an undefined opcode is encountered. The action of the CPU in response to Trap is to jump to address 00000000H with the status bit(s) set. This response is similar to the Z180 MPU's action on execution of an undefined instruction. The Trap is enabled immediately after reset, and it is not maskable. This feature can be used to increase software reliability or to implement "extended" instructions. An undefined opcode can be fetched from the instruction stream, or it can be returned as a vector in an interrupt acknowledge transaction in Interrupt mode 0.

Since it jumps to address 00000000H, it is necessary to have a Trap handling routine at the beginning of the program if processing is to proceed. Otherwise, it behaves just like a reset for the CPU. For a detailed description, refer to Chapter 6.

5.5 INSTRUCTION SET FUNCTIONAL GROUPS

This section presents an overview of the Z380 instruction set, arranged by functional groups. (See Section 5.5 for an explanation of the notation used in Tables 5-2 through 5-11).

5.5.1 8-Bit Load/Exchange Group

This group of instructions (Table 5-2) includes load instructions for transferring data between byte registers, transferring data between a byte register and memory, and loading immediate data into byte register or memory. For the supported source/destination combinations, refer to Table 5-3.

An Exchange instruction is available for swapping the contents of the accumulator with another register or with memory, as well as between registers. Also, exchange instructions are available which swap the contents of the register in the primary register bank and auxiliary register bank.

The instruction in this group does not affect the flags.

Table 5-2. 8-Bit Load Group Instructions

Instruction Name	Format	Note
Exchange with Accumulator	EX A,r	
	EX A,(HL)	
Exchange r and r'	EX r,r'	r=A, B, C, D, E, H or L
Load Accumulator	LD A,src	See Table 5-3
	LD dst,A	See Table 5-3
Load Immediate	LD dst,n	See Table 5-3
	LD (HL),n	See Table 5-3
Load Register (Byte)	LD R,src	See Table 5-3
	LD R,(HL)	See Table 5-3
	LD dst,R	See Table 5-3
	LD (HL),R	See Table 5-3

Table 5-3. 8-Bit Load Group Allowed Source/Destination Combinations

Source

Dist.	A	B	C	D	E	H	L	IXH	IXL	IYH	IYL	n	(nn)	(BC)	(DE)	(HL)	(IX+d)	(IY+d)
A	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
B	√	√	√	√	√	√	√	√	√	√	√	√				√	√	√
C	√	√	√	√	√	√	√	√	√	√	√	√				√	√	√
D	√	√	√	√	√	√	√	√	√	√	√	√				√	√	√
E	√	√	√	√	√	√	√	√	√	√	√	√				√	√	√
H	√	√	√	√	√	√	√					√				√	√	√
L	√	√	√	√	√	√	√					√				√	√	√
IXH	√	√	√	√	√			√	√			√						
IXL	√	√	√	√	√			√	√			√						
IYH	√	√	√	√	√					√	√	√						
IYL	√	√	√	√	√					√	√	√						
(BC)	√																	
(DE)	√																	
(HL)	√	√	√	√	√	√	√					√						
(nn)	√																	
(IX+d)	√	√	√	√	√	√	√					√						
(IY+d)	√	√	√	√	√	√	√					√						

Note: √ are supported combinations.

5.5.2 16-Bit and 32-Bit Load, Exchange, SWAP, and PUSH/POP Group

This group of load, exchange, and PUSH/POP instructions (Table 5-4) allows one or two words of data (two bytes equal one word) to be transferred between registers and memory.

The exchange instructions (Table 5-5) allow for switching between the primary and alternate register files, exchanging the contents of two register files, exchanging the contents of an addressing register with the top word on the stack. For possible combinations of the word exchange instructions, refer to Table 5-5. The 16-bit and 32-bit loads include transfer between registers and memory and immediate loads of registers or memory. The Push and Pop stack instructions are also included in this group. None of these instructions affect the CPU flags, except for EX AF, AF'.

Table 5-6 has the supported source/destination combination for the 16-bit and 32-bit load instructions. The transfer size, 16-bit or 32-bit, is determined by the status of LW bit in SR, or by DDIR Decoder Directives.

PUSH/POP instructions are used to save/restore the contents of a register onto the stack. It can be used to exchange data between procedures, save the current register file on context switching, or manipulate data on the stack, such as return addresses. Supported sources are listed in Table 5-7.

Swap instructions allows swapping of the contents of the Word wide register (BC, DE, HL, IX, or IY) with its Extended portion. These instructions are useful to manipulate the upper word of the register to be set in Word mode. For example, when doing data accesses, other than 00000000H-0000FFFFH address range, use this instruction to set "data frame" addresses.

This group of instructions is affected by the status of the LW bit in SR (Select Register), and Decoder Directives which specifies the operation mode in Word or Long Word.

Table 5-4. 16-Bit and 32-Bit Load, Exchange, PUSH/POP Group Instructions

Instruction Name	Format	Note
Exchange Word/Long Word Registers	EX dst,src	See Table 5-5
Exchange Byte/Word Registers with Alternate Bank	EXX	
Exchange Register Pair with Alternate Bank	EX RR,RR'	RR = AF, BC, DE, or HL
Exchange Index Register with Alternate Bank	EXXX	
	EXXY	
Exchange All Registers with Alternate Bank	EXALL	
Load Word/Long Word Registers	LD dst,src	See Table 5-6
	LDW dst,src	See Table 5-6
POP	POP dst	See Table 5-7
PUSH	PUSH src	See Table 5-7
Swap Contents of D31-D16 and D15-D0	SWAP dst	dst = BC, DE, HL, IX, or IY

Table 5-5. Supported Source and Destination Combination for 16-Bit and 32-Bit Exchange Instructions

	Source				
Destination	**BC**	**DE**	**HL**	**IX**	**IY**
BC	√	√	√	√	
DE		√	√	√	
HL			√	√	
IX				√	
(SP)		√	√	√	

Note: √ are supported combinations. The exchange instructions which designate IY register as destination are covered by the other combinations. These Exchange Word instructions are affected by Long Word mode.

5.5.2 16-Bit and 32-Bit Load, Exchange, SWAP and PUSH/POP Group (Continued)

Table 5-6. Supported Source and Destination Combination for 16-Bit and 32-Bit Load Instructions.

	Source													
Destination	**BC**	**DE**	**HL**	**IX**	**IY**	**SP**	**nn**	**(nn)**	**(BC)**	**(DE)**	**(HL)**	**(IX+d)**	**(IY+d)**	**(SP+d)**
BC	L	L	L	L	L		IL	IL	L	L	L	IL	IL	IL
DE	L	L	L	L	L		IL	IL	L	L	L	IL	IL	IL
HL	L	L	L	L	L		IL	IL	L	L	L	IL	IL	IL
IX	L	L	L		L		IL	IL	L	L	L		IL	IL
IY	L	L	L	L			IL	IL	L	L	L	IL		IL
SP			L	L	L		IL	IL						
(BC)	L	L	L	L	L		ILW							
(DE)	L	L	L	L	L		ILW							
(HL)	L	L	L	L	L		ILW							
(nn)	IL	IL	IL	IL	IL	IL								
(IX+d)	IL	IL	IL		IL									
(IY+d)	IL	IL	IL	IL										
(SP+d)	IL	IL	IL	IL	IL									

Note: The column with the character(s) are the allowed source/destination combinations. The combination with "L" means that the instruction is affected by Long Word mode, "I" means that the instruction is can be used with DDIR Immediate instruction. Also, "W" means the instruction uses the mnemonic of "LDW" instead of "LD".

Table 5-7. Supported Operand for PUSH/POP Instructions

	AF	**BC**	**DE**	**HL**	**IX**	**IY**	**SR**	**nn**
PUSH	√	√	√	√	√	√	√	√
POP	√	√	√	√	√	√	√	

Note: These PUSH/POP instructions are affected by Long Word mode of operations.

5.5.3 Block Transfer and Search Group

This group of instructions (Table 5-8) supports block transfer and string search functions. Using these instructions, a block of up to 65536 bytes of byte, Word, or Long Word data can be moved in memory, or a byte string can be searched until a given value is found. All the operations can proceed through the data in either direction. Furthermore, the operations can be repeated automatically while decrementing a length counter until it reaches zero, or they can operate on one storage unit per execution with the length counter decremented by one and the source and destination pointer register properly adjusted. The latter form is useful for implementing more complex operations in software by adding other instructions within a loop containing the block instructions.

Various Z380 CPU registers are dedicated to specific functions for these instructions—the BC register for a counter, the DEz/DE and HLz/HL registers for memory pointers, and the accumulator for holding the byte value being sought. The repetitive forms of these instructions are interruptible; this is essential since the repetition count can be as high as 65536. The instruction can be interrupted after any interaction, in which case the address of the instruction itself, rather than next one, is saved on the stack. The contents of the operand pointer registers, as well as the repetition counter, are such that the instruction can simply be reissued after returning from the interrupt without any visible difference in the instruction execution.

In case of Word or Long Word block transfer instructions, the counter value held in the BC register is decremented by two or four, depending on the LW bit status. Since exiting from these instructions will be done when counter value gets to 0, the count value stored in the BC registers has to be an even number (D0 = 0) in Word mode transfer, and a multiple of four in Long Word mode (D1 and D0 are both 0). Also, in Word or Long Word Block transfer, memory pointer values are recommended to be even numbers so the number of the transactions will be minimized.

Note that regardless of the Z380's operation mode, Native or Extended, memory pointer increment/decrement will be done in modulo 2^{32}. For example, if the operation is LDI and HL31-HL0 (HLz and HL) hold 0000FFFF, after the operation the value in the HL31-HL0 will be 00010000.

Table 5-8. Block Transfer and Search Group

Instruction Name	Format
Compare and Decrement	CPD
Compare, Decrement and Repeat	CPDR
Compare and Increment	CPI
Compare, Increment and Repeat	CPIR
Load and Decrement	LDD
Load , Decrement and Repeat	LDDI
Load and Increment	LDI
Load, Increment and Repeat	LDIR
Load and Decrement in Word/Long Word	LDDW
Load, Decrement and Repeat in Word/Long Word	LDDRW
Load and Increment in Word/Long Word	LDIW
Load, Increment and Repeat in Word/Long Word	LDIRW

5.5.4 8-bit Arithmetic and Logical Group

This group of instructions (Table 5-9) perform 8-bit arithmetic and logical operations. The Add, Add with Carry, Subtract, Subtract with Carry, AND, OR, Exclusive OR, and Compare takes one input operand from the accumulator and the other from a register, from immediate data in the instruction itself, or from memory. For memory addressing modes, follows are supported—Indirect Register, Indexed, and Direct Address—except multiplies, which returns the 16-bit result to the same register by multiplying the upper and lower bytes of one of the register pair (BC, DE, HL, or SP).

The Increment and Decrement instructions operate on data in a register or in memory; all memory addressing modes are supported. These instructions operate only on the accumulator—Decimal Adjust, Complement, and Negate. The final instruction in this group, Extend Sign, sets the CPU flags according to the computed result.

The EXTS instruction extends the sign bit and leaves the result in the HL register. If it is in Long Word mode, HLz (HL31-HL16) portion is also affected.

The TST instruction is a nondestructive AND instruction. It ANDs "A" register and source, and changes flags according to the result of operation. Both source and destination values will be preserved.

Table 5-9. Supported Source/Destination for 8-Bit Arithmetic and Logic Group

Instruction Name	Format	src/ dst	A	B	C	D	E	H	L	IXH	IXL	IYH	IYL	n	(HL)	(IX+d)	(IY+x)
Add With Carry (Byte)	ADC A,src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Add (Byte)	ADD A,src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
AND	AND [A,]src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Compare (Byte)	CP [A,]src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Complement Accumulator	CPL [A]	dst	√														
Decimal Adjust Accumulator	DAA	dst	√														
Decrement (Byte)	DEC dst	dst	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Extend Sign (Byte)	EXTS [A]	dst	√														
Increment (Byte)	INC dst	dst	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Multiply (Byte)	MLT src	Note 1															
Negate Accumulator	NEG [A]	dst	√														
OR	OR [A,]src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Subtract with Carry (Byte)	SBC A,src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Subtract (Byte)	SUB [A,]src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Nondestructive Test	TST dst	src	√	√	√	√	√	√	√					√	√		
Exclusive OR	XOR [A,]src	src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

Note 1: dst = BC, DE, HL, or SP.

5.5.5 16-Bit Arithmetic Operation

This group of instructions (Table 5-10) provide 16-bit arithmetic instructions. The Add, Add with Carry, Subtract, Subtract with Carry, AND, OR, Exclusive OR, and Compare takes one input operand from an addressing register and the other from a 16-bit register, or from the instruction itself; the result is returned to the addressing register. The 16-bit Increment and Decrement instructions operate on data found in a register or in memory; the Indirect Register or Direct Address addressing mode can be used to specify the memory operand.

The remaining 16-bit instructions provide general arithmetic capability using the HL register as one of the input operands. The word Add, Subtract, Compare, and signed and unsigned Multiply instructions take one input operand from the HL register and the other from a 16-bit register, from the instruction itself, or from memory using Indexed or Direct Address addressing mode. The 32-bit result of a multiply is returned to the HLz and HL (HL31-HL0). The unsigned divide instruction takes a 16-bit dividend from the HL register and a 16-bit divisor from a register, from the instruction, or memory using the Indexed mode. The 16-bit quotient is returned in the HL register and the 16-bit reminder is returned to the HLz (HL31-HL16). The Extend Sign instruction takes the contents of the HL register and delivers the 32-bit result to the HLz and HL registers. The Negate HL instruction negates the contents of the HL register.

Except for Increment, Decrement, and Extend Sign, all the instructions in this group set the CPU flags to reflect the computed result.

Table 5-10. 16-Bit Arithmetic Operation

Instruction Name	Format	src/ dst	BC	DE	HL	SP	IX	IY	nn	(nn)	(IX+d)	(IY+d)	
Add With Carry (Word)	ADC HL,src	src	√	√	√	√							
	ADCW [HL],src	src	√	√	√		√	√	√		√	√	
Add (Word)	ADD HL,src	src	√	√	√	√				√			X
	ADD IX,src	src	√	√		√	√						X
	ADD IY,src	src	√	√		√		√					X
	ADDW [HL,]src	src	√	√	√		√	√	√		√	√	
Add to Stack Pointer	ADD SP,nn	src							√				X
AND Word	ANDW [HL,]src	src	√	√	√		√	√	√		√	√	
Complement Accumulator	CPLW [HL]	dst			√								
Compare (Word)	CPW [HL,]src	src	√	√	√		√	√	√		√	√	
Decrement (Word)	DEC[W] dst	dst	√	√	√	√	√	√					X
Divide Unsigned	DIVUW [HL,]src	src	√	√	√		√	√	√		√	√	
Extend Sign (Word)	EXTSW [HL]	dst			√								
Increment (Word)	INC[W] dst	dst	√	√	√	√	√	√					X
Multiply Word Signed	MULT [HL,]src	src	√	√	√		√	√	√		√	√	
Multiply Word Unsigned	MULTUW [HL,]src	src	√	√	√		√	√	√		√	√	
Negate Accumulator	NEGW [A]	dst			√								
OR Word	ORW [HL,]src	src	√	√	√		√	√	√		√	√	
Subtract with Carry (Word)	SBC HL,src	src	√	√	√	√				√			
	SBCW [HL],src	src	√	√	√		√	√	√		√	√	
Subtract (Word)	SUB HL,(nn)	src								√			X
	SUBW [HL,]src	src	√	√	√		√	√	√		√	√	
Subtract from Stack Pointer	SUB SP,nn	src							√				X
Exclusive OR	XORW [HL,]src	src	√	√	√		√	√	√		√	√	

Note: that the instructions with "X" at the rightmost column is affected by Extended mode. These operate across all the 32 bits in Modulo 2^{32} for address calculation.

5.5.6 8-Bit Manipulation, Rotate and Shift Group

Instructions in this group (Table 5-11) test, set, and reset bits within bytes, and rotate and shift byte data one bit position. Bits to be manipulated are specified by a field within the instruction. Rotate can optionally concatenate the Carry flag to the byte to be manipulated. Both left and right shifting is supported. Right shifts can either shift 0 into bit 7 (logical shifts), or can replicate the sign in bits 6 and 7 (arithmetic shifts). All these instructions, Set Bit and Reset Bit, set the CPU flags according to the calculated result; the operand can be a register or a memory location specified by the Indirect Register or Indexed addressing mode.

The RLD and RRD instructions are provided for manipulating strings of BCD digits; these rotate 4-bit quantities in memory specified by the Indirect Register. The low-order four bits of the accumulator are used as a link between rotation of successive bytes.

Table 5-11. Bit Set/Reset/Test, Rotate and Shift Group

Instruction Name	Format	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
Bit Test	BIT dst	√	√	√	√	√	√	√	√	√	√
Reset Bit	RES dst	√	√	√	√	√	√	√	√	√	√
Rotate Left	RL dst	√	√	√	√	√	√	√	√	√	√
Rotate Left Accumulator	RLA	√									
Rotate Left Circular	RLC dst	√	√	√	√	√	√	√	√	√	√
Rotate Left Circular (Accumulator)	RLCA	√									
Rotate Left Digit	RLD	√									
Rotate Right	RR dst	√	√	√	√	√	√	√	√	√	√
Rotate Right Accumulator	RRA	√									
Rotate Right Circular	RRC dst	√	√	√	√	√	√	√	√	√	√
Rotate Right Circular (Accumulator)	RRCA	√									
Rotate Right Digit	RRD	√									
Set Bit	SET dst	√	√	√	√	√	√	√	√	√	√
Shift Left Arithmetic	SLA dst	√	√	√	√	√	√	√	√	√	√
Shift Right Arithmetic	SRA dst	√	√	√	√	√	√	√	√	√	√
Shift Right Logical	SRL	√	√	√	√	√	√	√	√	√	√

5.5.7 16-Bit Manipulation, Rotate and Shift Group

Instructions in this group (Table 5-12) rotate and shift word data one bit position. Rotate can optionally concatenate the Carry flag to the word to be manipulated. Both left and right shifting is supported. Right shifts can either shift 0 into bit 15 (logical shifts), or can replicate the sign in bits 14 and 15 (arithmetic shifts). The operand can be a register pair or memory location specified by the Indirect Register or Indexed addressing mode, as shown below.

Table 5-12. 16-Bit Rotate and Shift Group.

		Destination								
Instruction Name	**Format**	**BC**	**DE**	**HL**	**IX**	**IY**	**(HL)**	**(HL)**	**(IX+d)**	**(IY+d)**
Rotate Left Word	RLW dst	√	√	√	√	√	√	√	√	√
Rotate Left Circular Word	RLCW dst	√	√	√	√	√	√	√	√	√
Rotate Right Word	RRW dst	√	√	√	√	√	√	√	√	√
Rotate Right Circular Word	RRCW dst	√	√	√	√	√	√	√	√	√
Shift Left Arithmetic Word	SLAW dst	√	√	√	√	√	√	√	√	√
Shift Right Arithmetic Word	SRAW dst	√	√	√	√	√	√	√	√	√
Shift Right Logical Word	SRLW	√	√	√	√	√	√	√	√	√

5.5.8 Program Control Group

This group of instructions (Table 5-13) affect the Program Counter (PC) and thereby control program flow. The CPU registers and memory are not altered except for the Stack Pointer and the Stack, which play a significant role in procedures and interrupts. (An exception is Decrement and Jump if Non-Zero [DJNZ], which uses a register as a loop counter.) The flags are also preserved except for the two instructions specifically designed to set and complement the Carry flag.

The Set/Reset Condition flag instructions can be used with Conditional Jump, conditional Jump Relative, Conditional Call, and Conditional Return instructions to control the program flow.

The Jump and Jump Relative (JR) instructions provide a conditional transfer of control to a new location if the processor flags satisfy the condition specified in the instruction. Jump Relative, with an 8-bit offset (JR e), is a two byte instruction that jumps any instructions within the range –126 to +129 bytes from the location of this instruction. Most conditional jumps in programs are made to locations only a few bytes away; the Jump Relative, with an 8-bit offset, exploits this fact to improve code compactness and efficiency. Jump Relative, with a 16-bit offset (JR [cc,]ee), is a four byte instruction that jumps any instructions within the range –32765 to +32770 bytes from the location of this instruction, and Jump Relative, with a 24-bit offset (JR [cc,] eee), is a five byte instruction that jumps any instructions within the range –8388604 to +8388611 bytes from the location of this instruction. By using these Jump Relative instructions with 16-bit or 24-bit offsets allows to write relocatable (or location independent) programs.

Call and Restart are used for calling subroutines; the current contents of the PC are pushed onto the stack and the effective address indicated by the instruction is loaded into the PC. The use of a procedure address stack in this manner allows straightforward implementation of nested and recursive procedures. Call, Jump, and Jump Relative can be unconditional or based on the setting of a CPU flag.

Call Relative (CALR) instructions work just like ordinary Call instructions, but with Relative address. An 8-bit, 16-bit, or 24-bit offset value can be used, and that allows to call procedure within the range of –126 to +129 bytes (8-bit offset;CALR [cc,]e), –32765 to +32770 bytes (16-bit offset; CALR [cc,]ee), or –8388604 to +8388611 bytes (JR [cc,] eee) are supported. These instructions are really useful to program relocatable programs.

Jump is available with Indirect Register mode in addition to Direct Address mode. It can be useful for implementing complex control structures such as dispatch tables. When using Direct Address mode for a Jump or Call, the operand is used as an immediate value that is loaded into the PC to specify the address of the next instruction to be executed.

The conditional Return instruction is a companion to the call instruction; if the condition specified in the instruction is satisfied, it loads the PC from the stack and pops the stack.

A special instruction, Decrement and Jump if Non-Zero (DJNZ), implements the control part of the basic Pascal FOR loop which can be implemented in an instruction. It supports 8-bit, 16-bit, and 24-bit displacement.

Note that Jump Relative, Call Relative, and DJNZ instructions use modulo 2^{16} in Native mode, and 2^{32} in Extended mode for address calculation. So it is possible that the Z380 CPU can jump to an unexpected address.

Table 5-13. Program Control Group Instructions

Instruction Name	Format	nn	(PC+d)	(HL)	(IX)	(IY)
Call	CALL cc,dst	√				
Complement Carry Flag	CCF					
Call Relative	CALR cc,dst		√			
Decrement and Jump if Non-zero	DJNZ dst		√			
Jump	JP cc,dst	√				
	JP dst			√	√	√
Jump Relative	JR cc,dst		√			
Return	RET cc					
Restart	RST p	√				
Set Carry Flag	SCF					

5.5.9 External Input/Output Instruction Group

This group of instructions (Table 5-14) are used for transferring a byte, a word, or string of bytes or words between peripheral devices and the CPU registers or memory. Byte I/O port addresses transfer bytes on D7-D0 only. These 8-bit peripherals in a 16-bit data bus environment must be connected to data line D7-D0. In an 8-bit data bus environment, word I/O instructions to external I/O peripherals should not be used; however, on-chip peripherals which is external to the CPU core and assigned as word I/O device can still be accessed by word I/O instructions.

The instructions for transferring a single byte (IN, OUT) can transfer data between any 8-bit CPU register or memory address specified in the instruction and the peripheral port specified by the contents of the C register. The IN instruction sets the CPU flags according to the input data; however, special instructions restricted to using the CPU accumulator and Direct Address mode and do not affect the CPU flags. Another variant tests an input port specified by the contents of the C register and sets the CPU flags without modifying CPU registers or memory.

The instructions for transferring a single word (INW, OUTW) can transfer data between the register pair and the peripheral port specified by the contents of the C register. For Word I/O, the contents of B, D, or H appear on D7-D0 and the contents of C, E, or L appear D15-D7. These instructions do not affect the CPU flags.

Also, there are I/O instructions available which allow to specify 16-bit absolute I/O address (with DDIR decoder directives, a 24-bit or 32-bit address is specified) is available. These instructions do not affect the CPU flags.

The remaining instructions in this group form a powerful and complete complement of instructions for transferring blocks of data between I/O ports and memory. The operation of these instructions is very similar to that of the block move instructions described earlier, with the exception that one operand is always an I/O port whose address remains unchanged while the address of the other operand (a memory location) is incremented or decremented. In Word mode of transfer, the counter (i.e., BC register) holds the number of transfers, rather than number of bytes to transfer in memory-to-memory word block transfer. Both byte and word forms of these instructions are available. The automatically repeating forms of these instructions are interruptible, like memory-to-memory transfer.

The I/O addresses output on the address bus is dependant on the I/O instruction, as listed in Table 2-1.

5.5.9 External Input/Output Instruction Group (Continued)

Table 5-14. External I/O Group Instructions.

Instruction Name	Format	
Input	IN dst,(C)	dst=A, B, C, D, E, H or L
Input Accumulator	IN A,(n)	
Input to Word-Wide Register	INW dst,(C)	dst=BC, DE or HL
Input Byte from Absolute Address	INAW A,(nn)	
Input Word from Absolute Address	INAW HL,(nn)	
Input and Decrement (Byte)	IND	
Input and Decrement (Word)	INDW	
Input, Decrement, and Repeat (Byte)	INDR	
Input, Decrement, and Repeat (Word)	INDRW	
Input and Increment (Byte)	INI	
Input and Increment (Word)	INIW	
Input, Increment, and Repeat (Byte)	INIR	
Input, Increment, and Repeat (Word)	INIRW	
Output	OUT (C),src	src = A, B, C, D, E, H, L, or n
Output Accumulator	OUT (n),A	
Output from Word-Wide Register	OUTW (C), src	src = BC, DE, HL, or nn
Output Byte from Absolute Address	OUTAW (nn),A	
Output Word from Absolute Address	OUTAW (nn),HL	
Output and Decrement (Byte)	OUTD	
Output and Decrement (Word)	OUTDW	
Output, Decrement, and Repeat (Byte)	OTDR	
Output, Decrement, and Repeat (Word)	OTDRW	
Output and Increment (Byte)	OUTI	
Output and Increment (Word)	OTIW	
Output, Increment, and Repeat (Byte)	OTIR	
Output, Increment, and Repeat (Word)	OTIRW	

5.5.10 Internal I/O Instruction Group

This group (Table 5-15) of instructions is used to access on-chip I/O addressing space on the Z380 CPU. This group consists of instructions for transferring a byte from/ to Internal I/O locations and the CPU registers or memory, or a blocks of bytes from the memory to the same size of Internal I/O locations for initialization purposes. These instructions are originally assigned as newly added I/O instructions on the Z180 MPU to access Page 0 I/O addressing space. There is 256 Internal I/O locations, and all of them are byte-wide. When one of these I/O instructions is executed, the Z380 MPU outputs the register address being accessed in a pseudo transaction of two BUSCLK durations cycle, with the address signals A31-A8 at 0. In the pseudo transactions, all bus control signals are at their inactive state.

The instructions for transferring a single byte (IN0, OUT0) can transfer data between any 8-bit CPU register and the Internal I/O address specified in the instruction. The IN0 instruction sets the CPU flags according to the input data; however, special instructions which do not have a destination in the instruction with Direct Address (IN0 (n)), do not affect the CPU register, but alters flags accordingly. Another variant, the TSTIO instruction, does a logical AND to the instruction operand with the internal I/O location specified by the C register and changes the CPU flags without modifying CPU registers or memory.

The remaining instructions in this group form a powerful and complete complement of instructions for transferring blocks of data from memory to Internal I/O locations. The operation of these instructions is very similar to that of the block move instructions described earlier, with the exception that one operand is always an Internal I/O location whose address also increments or decrements by one automatically, Also, the address of the other operand (a memory location) is incremented or decremented. Since Internal I/O space is byte-wide, only byte forms of these instructions are available. Automatically repeating forms of these instructions are interruptible, like memory-to-memory transfer.

Table 5-15. Internal I/O Instruction Group

Instruction Name	Format	
Input from Internal I/O Location	IN0 dst,(n)	dst=A, B, C, D, E, H or L
Input from Internal I/O Location(Nondestructive)	IN0 (n)	
Test I/O	TSTIO n	
Output to Internal I/O Location	OUT0 (n),src	src=A, B, C, D, E, H or L
Output to Internal I/O and Decrement	OTDM	
Output to Internal I/O and Increment	OTIM	
Output to Internal I/O, Decrement and Repeat	OTDMR	
Output to Internal I/O, Increment and Repeat	OTIMR	

Currently, the Z380 CPU core has the following registers as a part of the CPU core:

Register Name	Internal I/O address
Interrupt Enable Register	16H
Assigned Vector Base Register	17H
Trap Register	18H
Chip Version ID Register	0FFH

Chip Version ID register returns one byte data, which indicates the version of the CPU, or the specific implementation of the Z380 CPU based Superintegration device. Currently, the value 00H is assigned to the Z380 MPU, and other values are reserved.

For the other three registers, refer to Chapter 6, "Interrupt and Trap."

Also, the Z380 MPU has registers to control chip selects, refresh, waits, and I/O clock divide to Internal I/O address 00H to 10H. For these register, refer to Z380 MPU Product specification.

5.5.11 CPU Control Group

The instructions in this group (Table 5-16) act upon the CPU control and status registers or perform other functions that do not fit into any of the other instruction groups. These include two instructions used for returning from an interrupt service routine. Return from Nonmaskable Interrupt (RETN) and Return from Interrupt (RETI) are used to pop the Program Counter from the stack and manipulate the Interrupt Enable Flag (IEF1 and IEF2), or to signal a reset to the Z80 peripherals family.

The Disable and Enable Interrupt instructions are used to set/reset interrupt mask. Without a mask parameters, it disables/enables maskable interrupt globally. With mask data, it enables/disables interrupts selectively.

HALT and SLEEP instructions stop the CPU and waits for an event to happen, or puts the system into the power save mode.

Bank Test instructions reports which register file, primary or alternate bank, is in use at the time, and reflect the status into a flag register. For example, this instruction is useful to implement the recursive program, which uses the alternate bank to save a register for the first time, and saves registers into memory thereafter.

Mode Test instructions reports the current mode of operation, Native/Extended, Word/Long Word, Locked or not. This instruction can be used to switch procedures depending on the mode of operation.

Load Accumulator from R or I Register instructions are used to report current interrupt mask status. Load from/to register instructions are used to initialize the I register.

Load Control register instructions are used to read/write the Status Register, set/reset control bit instructions and to set/reset the control bits in the SR.

The No Operation instruction does nothing, and can be used as a filler, for debugging purposes, or for timing adjustment.

Table 5-16. **CPU Control Group**

Instruction Name	Format	
Bank Test	BTEST	
Disable Interrupt	DI [mask]	
Enable Interrupt	EI [mask]	
HALT	HALT	
Interrupt Mode Select	IM p	
Load Accumulator from I or R Register	LD A,src	
Load I or R Register from Accumulator	LD dst,A	
Load I Register from HL Register	LD[W] HL,I	
Load HL Register from I Register	LD[W] HL,I	
Load Control	LDCTL dst,src	
Mode Test	MTEST	
No Operation	NOP	
Return from Interrupt	RETI	
Return from Nonmaskable Interrupt	RETN	
Reset Control Bit	RESC dst	dst=LCK, LW
Set Control Bit	SETC dst	dst=LCK, LW, XM
Sleep	SLP	

5.5.12 Decoder Directives

The Decoder Directives (Table 5-17) are a special instructions to expand the Z80 instruction set to handle the Z380's 4 Gbytes of linear memory addressing space. For details on this instruction, refer to Chapter 3.

Table 5-17. Decoder Directive Instructions

DDIR W	Word Mode
DDIR IB,W	Immediate Byte, Word Mode
DDIR IW,W	Immediate Word, Word Mode
DDIR IB	Immediate Byte
DDIR LW	Long Word Mode
DDIR IB,LW	Immediate Byte, Long Word Mode
DDIR IW,LW	Immediate Word, Long Word Mode
DDIR IW	Immediate Word

5.6 NOTATION AND BINARY ENCODING

The rest of this chapter consists of a detailed description of the Z380 CPU instructions, arranged in alphabetical order by mnemonic. This section describes the notational conventions used in the instruction descriptions and the binary encoding for register fields within the instruction's operation codes (opcodes).

The description of each instruction begins on a new page. The instruction mnemonic and name are printed in bold letters at the top of each page to enable the reader to easily locate a desired description. The assembly language syntax is then given in a single generic form that covers all the variants of the instruction, along with a list of applicable addressing modes. This is followed by a description of the operation performed by the instruction in "pseudo Pascal" fashion, a detailed description, a listing of all the flags that are affected by the instruction, and illustrations of the opcodes for all variants of the instruction.

Symbols. The following symbols are used to describe the instruction set.

n	An 8-bit constant
nn	A 16-bit constant
d	An 8-bit offset. (two's complement)
src	Source of the instruction
dst	Destination of the instruction
SR	Select Register
R	Any register. In Word operation, any register pair. Any 8-bit register (A, B, C, D, E, H, or L) for Byte operation.
IR	Indirect register
RX	Indexed register (IX or IY) in Word operation, IXH, IXL, IYH, or IYL for Byte operation.
SP	Current Stack Pointer
(C)	I/O Port pointed by C register
cc	Condition Code
[ ]	Optional field
( )	Indirect Address Pointer or Direct Address

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location.

The symbol "↔" indicates that the source and destination is swapping. For example,

dst ↔ src

indicates that the source data is swapped with the data in the destination; after the operation, data at "src" is in the "dst" location, and data in "dst " is in the "src" location.

The notation "dst (b)" is used to refer to bit "b" of a given location, "dst(m-n)" is used to refer to bit location m to n of the destination. For example,

HL(7) specifies bit 7 of the destination.
and
HL(23-16) specifies bit location 23 to 16 of the HL register.

Flags. The F register contains the following flags followed by symbols.

S	Sign Flag
Z	Zero Flag
H	Half Carry Flag
P/V	Parity/Overflow Flag
N	Add/Subtract Flag
C	Carry Flag

5.6 NOTATION AND BINARY ENCODING (Continued)

Condition Codes. The following symbols describe the condition codes.

Z	Zero*
NZ	Not Zero*
C	Carry*
NC	No Carry*
S	Sign
NS	No Sign
NV	No Overflow
V	Overflow
PE	Parity Even
PO	Parity Odd
P	Positive
M	Minus

*Abbreviated set

Field Encoding. For opcode binary format in the Tables, use the following convention:

For example, to get the opcode format on the instruction LD (IX+12h), C

First, find out the entry for "LD (XY+d),R". That entry has a opcode format of

11 y11 101   01 110 -r-   ← d →

On the bottom of the each instruction, there are the field encodings, if applicable. For the cases which call out "per convention," then use the following encoding:

r	Reg
000	B
001	C
010	D
011	E
100	H
101	L
111	A

To form the opcode, first, look for the "y" field value for IX register, which is 0.

Then find "r" field value for the C register, which is 001. Replace "y" and "r" field with the value from the table, replace "d" value with the real number. The results being:

76 543 210	HEX
11 011 101	DD
01 110 001	71
00 010 010	21

5.7 EXECUTION TIME

Table 5-18 details the execution time for each instruction encoding. All execution times are for instruction execution only. Clock cycles required for fetch and decode are not included because most of the time the clocks required for these operations occur in parallel with execution of the previous instruction(s).

r in the execution time column indicates a memory read operation. The time required for a read operation is shown in the Table 5-18 below.

w in the execution time column indicates a memory write operation. The time required for a write operation is shown in the Table 5-18 below.

i in the execution time column indicates an I/O read operation. The time required for a read operation is shown in the Table 5-18 below.

o in the execution time column indicates an I/O write operation. The time required for a write operation is shown in the Table 5-18 below.

All entries in the table below assume no wait states. The number of wait states per operation must be added to these numbers.

Table 5-18. Execution Time

Operation	Byte	Word	Word	Long	Long	Long	Long	Long
Sequence	B	W	B/B	W/W	W/B/B	B/W/B	B/B/W	B/B/B/B
Memory Read	3-4	3-4	5-6	5-6	7-8	7-8	7-8	9-10
Memory Write	0-1	0-1	2-3	2-3	4-5	4-5	4-5	6-7
Internal I/O Read	3-4	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Internal I/O Write	0-1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1X External I/O Read	4-5	4-5	N/A	N/A	N/A	N/A	N/A	N/A
1X External I/O Write	1-2	1-2	N/A	N/A	N/A	N/A	N/A	N/A
2X External I/O Read	9-11	9-11	N/A	N/A	N/A	N/A	N/A	N/A
2X External I/O Write	1-3	1-3	N/A	N/A	N/A	N/A	N/A	N/A
4X External I/O Read	17-21	17-21	N/A	N/A	N/A	N/A	N/A	N/A
4X External I/O Write	1-5	1-5	N/A	N/A	N/A	N/A	N/A	N/A
6X External I/O Read	25-31	25-31	N/A	N/A	N/A	N/A	N/A	N/A
6X External I/O Write	1-7	1-7	N/A	N/A	N/A	N/A	N/A	N/A
8X External I/O Read	33-41	33-41	N/A	N/A	N/A	N/A	N/A	N/A
8X External I/O Write	1-9	1-9	N/A	N/A	N/A	N/A	N/A	N/A

Note: Units are in Clocks. "N/A" is not applicable for that particular transaction.

ADC
ADD WITH CARRY (BYTE)

ADC A,src   src = R, RX, IM, IR, X

Operation: A ← A + src + C

The source operand together with the Carry flag is added to the accumulator and the sum is stored in the accumulator. The contents of the source is unaffected. Two's complement addition is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a carry from bit 3 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if both operands cleared otherwise
N: Cleared
C: Set if there is a carry from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	ADC A,R	10001-r-	2	
RX:	ADC A,RX	11y11101 1000110w	2	
IM:	ADC A,n	11001110 —n—	2	
IR:	ADC A,(HL)	10001110	2+r	
X:	ADC A,(XY+d)	11y11101 10001110—d—	4+r	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

ADC
ADD WITH CARRY (WORD)

ADC HL,src   dst = HL
src = BC, DE, HL, SP

Operation: HL(15-0) ← HL(15-0) + src(15-0) + C

The source operand together with the Carry flag is added to the HL register and the sum is stored in the HL register. The contents of the source are unaffected. Two's complement addition is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a carry from bit 11 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise
N: Cleared
C: Set if there is a carry from the most significant bit of the result; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	ADC HL,R	11101101 01rr1010	2	

Field Encodings: rr: 00 for BC, 01 for DE, 10 for HL, 11 for SP

ADCW
ADD WITH CARRY (WORD)

ADCW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) ← HL(15-0) + src(15-0) + C

The source operand together with the Carry flag is added to the HL register and the sum is stored in the HL register. The contents of the source are unaffected. Two's complement addition is performed.

Flags:
S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a carry from bit 11 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise
N: Cleared
C: Set if there is a carry from the most significant bit of the result; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	ADCW [HL,]R	11101101 100011rr	2	
RX:	ADCW [HL,]RX	11y11101 10001111	2	
IM:	ADCW [HL,]nn	11101101 10001110 -n(low)- n(high)-	2	
X:	ADCW [HL,](XY+d)	11y11101 11001110 ——d——	4+r	I

Field Encodings:
rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

ADD
ADD (BYTE)

ADD A,src src = R, RX, IM, IR, X

Operation: A ← A + src

The source operand is added to the accumulator and the sum is stored in the accumulator. The contents of the source are unaffected. Two's complement addition is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a carry from bit 3 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise
N: Cleared
C: Set if there is a carry from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	ADD A,R	10000-r-	2	
RX:	ADD A,RX	11y11101 1000010w	2	
IM:	ADD A,n	11000110 ——n—	2	
IR:	ADD A,(HL)	10000110	2+r	
X:	ADD A,(XY+d)	11y11101 10000110 ——d—	4+r	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

ADD
ADD (WORD)

ADD dst,src — dst = HL; src = BC, DE, HL, SP, DA
or
dst = IX; src = BC, DE, IX, SP
or
dst = IY; src = BC, DE, IY, SP

Operation:

```
If (XM) then begin
dst(31-0)  ←  dst(31-0) + src(31-0)
end
else begin
dst(15-0)  ←  dst(15-0) + src(15-0)
end
```

The source operand is added to the destination and the sum is stored in the destination. The contents of the source are unaffected. Two's complement addition is performed. Note that the length of the operand is controlled by the Extended/Native mode selection, which is consistent with the manipulation of an address by the instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Set if there is a carry from bit 11 of the result; cleared otherwise
V: Unaffected
N: Cleared
C: Set if there is a carry from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	ADD HL,R	00rr1001	2	X
RX:	ADD XY,R	11y11101 00rr1001	2	X
DA:	ADD HL,(nn)	11101101 11000110 -n(low)- n(high)-	2+r	I, X

Field Encodings: rr: 00 for BC, 01 for DE, 10 for register to itself, 11 for SP
y: 0 for IX, 1 for IY

ADD
ADD TO STACK POINTER (WORD)

ADD SP,src src = IM

Operation:

```
if (XM) then begin
    SP(31-0)    ←    SP(31-0) + src(31-0)
    end
else begin
    SP(15-0)    ←    SP(15-0) + src(15-0)
end
```

The source operand is added to the SP register and the sum is stored in the SP register. This has the effect of allocating or allocating space on the stack. Two's complement addition is performed.

Flags:

S: Unaffected
Z: Unaffected
H: Set if there is a carry from bit 11 of the result; cleared otherwise
V: Unaffected
N: Cleared
C: Set if there is a carry from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
IM:	ADD SP,nn	11101101 10000010 -n(low)- -n(high)	2	I, X

5

ADDW
ADD (WORD)

ADDW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) ← HL(15-0) + src(15-0)

The source operand is added to the HL register and the sum is stored in the HL register. The contents of the source are unaffected. Two's complement addition is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a carry from bit 11 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise
N: Cleared
C: Set if there is a carry from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	ADDW [HL,]R	11101101 100001rr	2	
RX:	ADDW [HL,]RX	11y11101 10000111	2	
IM:	ADDW [HL,]nn	11101101 10000110 -n(low)- n(high)-	2	
X:	ADDW [HL,](XY+d)	11y11101 11000110 —d—	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

AND
AND (BYTE)

AND [A,]src   src = R, RX, IM, IR, X

Operation: A ← A AND src

A logical AND operation is performed between the corresponding bits of the source operand and the accumulator and the result is stored in the accumulator. A 1 is stored wherever the corresponding bits in the two operands are both 1s; otherwise a 0 is stored. The contents of the source are unaffected.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if all bits of the result are zero; cleared otherwise
H: Set
P: Set if the parity is even; cleared otherwise
N: Cleared
C: Cleared

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	AND [A,]R	10100-r-	2	
RX:	AND [A,]RX	11y11101 1010010w	2	
IM:	AND [A,]n	11100110 ——n—	2	
IR:	AND [A,](HL)	10100110	2+r	
X:	AND [A,](XY+d)	11y11101 10100110——d—	4+r	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

ANDW
AND (WORD)

ANDW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) ← HL(15-0) AND src(15-0)

A logical AND operation is performed between the corresponding bits of the source operand and the HL register and the result is stored in the HL register. A 1 is stored wherever the corresponding bits in the two operands are both 1s; otherwise a 0 is stored. The contents of the source are unaffected.

Flags:
S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if all bits of the result are zero; cleared otherwise
H: Set
P: Set if the parity is even; cleared otherwise
N: Cleared
C: Cleared

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	ANDW [HL,]R	11101101 101001rr	2	
RX:	ANDW [HL,]RX	11y11101 10100111	2	
IM:	ANDW [HL,]nn	1110110110100110 n(low)- n(high)-	2	
X:	ANDW [HL,](XY+d)	11y11101 11100110 ——d—	4+r	I

Field Encodings:
rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

BIT
BIT TEST

BIT b,dst dst = R, IR, X

Operation: Z ← NOT dst(b)

The specified bit b within the destination operand is tested, and the Zero flag is set to 1 if the specified bit is 0, otherwise the Zero flag is cleared to 0. The contents of the destination are unaffected. The bit to be tested is specified by a 3-bit field in the instruction; this field contains the binary encoding for the bit number to be tested. The bit number b must be between 0 and 7.

Flags:

S: Unaffected
Z: Set if the specified bit is zero; cleared otherwise
H: Set
V: Unaffected
N: Cleared
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	BIT b,R	11001011 01bbb-r-	2	
IR:	BIT b,(HL)	11001011 01bbb110	2+r	
X:	BIT b,(XY+d)	11y11101 11001011 ——d— 01bbb110	4+r	I

Field Encodings: r: per convention
y: 0 for IX, 1 for IY

BTEST
BANK TEST

BTEST

Operation:

S ← SR(16)
Z ← SR(24)
V ← SR(0)
C ← SR(8)

The Alternate Register bits in the Select Register (SR) are transferred to the flags. This allows the program to determine the state of the machine.

Flags:

S: Set if the alternate bank IX is in use; cleared otherwise
Z: Set if the alternate bank IY is in use; cleared otherwise
H: Unaffected
V: Set if the alternate bank AF is in use; cleared otherwise
N: Unaffected
C: Set if the alternate bank of BC, DE and HL is in use; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	BTEST	11101101 11001111	2	

CALL
CALL

CALL [cc,]dst dst = DA

Operation:

```
if (cc is TRUE) then begin
    if (XM) then begin
        SP          ←   SP - 4
        (SP)        ←   PC(7-0)
        (SP+1)      ←   PC(15-8)
        (SP+2)      ←   PC(23-16)
        (SP+3)      ←   PC(31-24)
        PC(31-0)    ←   dst(31-0)
    else begin
        SP          ←   SP - 2
        (SP)        ←   PC(7-0)
        (SP+1)      ←   PC(15-8)
        PC(15-0)    ←   dst(15-0)
        end
    end
```

A conditional Call transfers program control to the destination address if the setting of a selected flag satisfies the condition code "cc" specified in the instruction; an Unconditional Call always transfers control to the destination address. The current contents of the Program Counter (PC) are pushed onto the top of the stack; the PC value used is the address of the first instruction byte following the Call instruction. The destination address is then loaded into the PC and points to the first instruction of the called procedure. At the end of a procedure a Return instruction (RET) can be used to return to the original program.

Each of the Zero, Carry, Sign, and Overflow Flags can be individually tested and a call performed conditionally on the setting of the flag.

The operand is not enclosed in parentheses with the CALL instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
DA:	CALL CC,addr	11-cc100 -a(low)- -a(high)	note	I, X
	CALL addr	11001101 -a(low)- -a(high)	4+w	I, X

Field Encodings: cc: 000 for NZ, 001 for Z, 010 for NC, 011 for C,
100 for PO or NV, 101 for PE or V, 110 for P or NS, 111 for M or S

Note: 2 if CC is false, 4+w if CC is true

CALR
CALL RELATIVE

CALR [cc,]dst   dst = RA

Operation:

```
if (cc is true) then begin
    dst            ←    SIGN EXTEND dst
    if (XM) then begin
        SP         ←    SP - 4
        (SP)       ←    PC(7-0)
        (SP+1)     ←    PC(15-8)
        (SP+2)     ←    PC(23-16)
        (SP+3)     ←    PC(31-24)
        PC(31-0)   ←    PC(31-0) + dst(31-0)
        end
    else begin
        SP         ←    SP - 2
        (SP)       ←    PC(7-0)
        (SP+1)     ←    PC(15-8)
        PC(15-0)   ←    PC(15-0) + dst(15-0)
        end
end
```

A conditional Call transfers program control to the destination address if the setting of a selected flag satisfies the condition code "cc" specified in the instruction; an unconditional call always transfers control to the destination address. The current contents of the Program Counter (PC) are pushed onto the top of the stack; the PC value used is the address of the first instruction byte following the Call instruction. The destination address is then loaded into the PC and points to the first instruction of the called procedure. At the end of a procedure a RETurn instruction is used to return to the original program. These instructions employ either an 8-bit, 16-bit, or 24-bit signed, two's complement displacement from the PC to permit calls within the range of -126 to +129 bytes, –32,765 to +32,770 bytes or –8,388,604 to +8,388,611 bytes from the location of this instruction.

Each of the Zero, Carry, Sign, and Overflow flags can be individually tested and a call performed conditionally on the setting of the flag.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
RA:	CALR CC,addr	11101101 11-cc100 —disp—	note	X
	CALR addr	11101101 11001101 —disp—	4+w	X
	CALR CC,addr	11011101 11-cc100 -d(low)- -d(high)	note	X
	CALR addr	11011101 11001101 -d(low)- -d(high)	4+w	X
	CALR CC,addr	11111101 11-cc100 -d(low)- -d(mid)- -d(high)	note	X
	CALR addr	11111101 11001101 -d(low)- -d(mid) -d(high)	4+w	X

Field Encodings: cc: 000 for NZ, 001 for Z, 010 for NC, 011 for C, 100 for PO or NV, 101 for PE or V, 110 for P or NS, 111 for M or S

Note: 2 if CC is false, 4+w if CC is true

CCF
COMPLEMENT CARRY FLAG

CCF

Operation: C ← NOT C

The Carry flag is inverted.

Flags:

S: Unaffected
Z: Unaffected
H: The previous state of the Carry flag
V: Unaffected
N: Cleared
C: Set if the Carry flag was clear before the operation; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	CCF	00111111	2	

CP
COMPARE (BYTE)

CP [A,]src   src = R, RX, IM, IR, X

Operation: A – src

The source operand is compared with the accumulator and the flags are set accordingly. The contents of the accumulator and the source are unaffected. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 4 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the operands are of different signs and the result is of the same sign as the source; cleared otherwise
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	CP [A,]R	10111-r-	2	
RX:	CP [A,]RX	11y11101 1011110w	2	
IM:	CP [A,]n	11111110 ——n—	2	
IR:	CP [A,](HL)	10111110	2+r	
X:	CP [A,](XY+d)	11y11101 10111110 ——d—	4+r	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

CPW
COMPARE (WORD)

CPW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) – src(15-0)

The source operand is compared with the HL register and the flags are set accordingly. The contents of the HL register and the source are unaffected. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 12 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the operands are of different signs and the result is of the same sign as the source; cleared otherwise
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	CPW [HL,]R	11101101 101111rr	2	
RX:	CPW [HL,]RX	11y11101 10111111	2	
IM:	CPW [HL,]nn	11101101 10111110 -n(low)- n(high)-	2	
X:	CPW [HL,](XY+d)	11y11101 11111110 ——d—	4+r	I

Field Encodings:

rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

CPD
COMPARE AND DECREMENT (BYTE)

CPD

Operation:

```
A - (HL)
if (XM) then begin
    HL(31-0)    ←    HL(31-0) - 1
    end
else begin
    HL(15-0)    ←    HL(15-0) - 1
    end
BC(15-0)        ←    BC(15-0) - 1
```

This instruction is used for searching strings of byte data. The byte of data at the location addressed by the HL register is compared with the contents of the accumulator and the Sign and Zero flags are set to reflect the result of the comparison. The contents of the accumulator and the memory bytes are unaffected. Two's complement subtraction is performed. Next the HL register is decremented by one, thus moving the pointer to the previous element in the string. The BC register, used as a counter, is then decremented by one.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero, indicating that the contents of the accumulator and the memory byte are equal; cleared otherwise
H: Set if there is a borrow from bit 4 of the result; cleared otherwise
V: Set if the result of decrementing BC is not equal to zero; cleared otherwise
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	CPD	11101101 10101001	3+r	X

CPDR
COMPARE, DECREMENT AND REPEAT (BYTE)

CPDR

Operation:

```
Repeat until (BC=0 OR match) begin
    A - (HL)
    if (XM) then begin
        HL(31-0)      ←      HL(31-0) - 1
        end
    else begin
        HL(15-0)      ←      HL(15-0) - 1
        end
    BC(15-0)          ←      BC(15-0) - 1
end
```

This instruction is used for searching strings of byte data. The bytes of data starting at the location addressed by the HL register are compared with the contents of the accumulator until either an exact match is found or the string length is exhausted becuase the BC register has decremented to zero. The Sign and Zero flags are set to reflect the result of the comparison. The contents of the accumulator and the memory bytes are unaffected.Two's complement subtraction is performed.

After each comparison, the HL register is decremented by one, thus moving the pointer to the previous element in the string.

The BC register, used as a counter, is then decremented by one. If the result of decrementing the BC register is not zero and no match has been found, the process is repeated. If the contents of the BC register are zero at the start of this instruction, a string length of 65,536 is indicated.

This instruction can be interrupted after each execution of the basic operation. The PC value at the start of this instruction is pushed onto the stack so that the instruction can be resumed.

Flags:

S: Set if the last result is negative; cleared otherwise
Z: Set if the last result is zero, indicating a match; cleared otherwise
H: Set if there is a borrow from bit 4 of the last result; cleared otherwise
V: Set if the result of decrementing BC is not equal to zero; cleared otherwise
N: Set
C: Unaffected

Addressing Mode

Syntax	Instruction Format	Execute Time	Note
CPDR	11101101 10111001	(3+r)n	X

CPI
COMPARE AND INCREMENT (BYTE)

CPI

Operation:

```
A - (HL)
if (XM) then begin
    HL(31-0)    <-    HL(31-0) + 1
    end
else begin
    HL(15-0)    <-    HL(15-0) + 1
    end
BC(15-0)        <-    BC(15-0) - 1
```

This instruction is used for searching strings of byte data. The byte of data at the location addressed by the HL register is compared with the contents of the accumulator and the Sign and Zero flags are set to reflect the result of the comparison. The contents of the accumulator and the memory bytes are unaffected. Two's complement subtraction is performed. Next the HL register is incremented by one, thus moving the pointer to the next element in the string. The BC register, used as a counter, is then decremented by one.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero, indicating that the contents of the accumulator and the memory byte are equal; cleared otherwise
H: Set if there is a borrow from bit 4 of the result; cleared otherwise
V: Set if the result of decrementing BC is not equal to zero; cleared otherwise
N: Set
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	CPI	11101101 10100001	3+r	X

CPIR
COMPARE, INCREMENT AND REPEAT (BYTE)

CPIR

Operation:

```
Repeat until (BC=0 OR match) begin
    A - (HL)
    if (XM) then begin
        HL(31-0)        ←       HL(31-0) + 1
        end
    else begin
        HL(15-0)        ←       HL(15-0) + 1
        end
    BC(15-0)            ←       BC(15-0) - 1
end
```

This instruction is used for searching strings of byte data. The bytes of data starting at the location addressed by the HL register are compared with the contents of the accumulator until either an exact match is found or the string length is exhausted becuase the BC register has decremented to zero. The Sign and Zero flags are set to reflect the result of the comparison. The contents of the accumulator and the memory bytes are unaffected. Two's complement subtraction is performed.

After each comparison, the HL register is incremented by one, thus moving the pointer to the next element in the string. The BC register, used as a counter, is then decremented by one. If the result of decrementing the BC register is not zero and no match has been found, the process is repeated. If the contents of the BC register are zero at the start of this instruction, a string length of 65,536 is indicated.

This instruction can be interrupted after each execution of the basic operation. The PC value at the start of this instruction is pushed onto the stack so that the instruction can be resumed.

Flags:

S: Set if the last result is negative; cleared otherwise
Z: Set if the last result is zero, indicating a match; cleared otherwise
H: Set if there is a borrow from bit 4 of the last result; cleared otherwise
V: Set if the result of decrementing BC is not equal to zero; cleared otherwise
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	CPIR	11101101 10110001	(3+r)n	X

CPL
COMPLEMENT ACCUMULATOR

CPL [A]

Operation: A ← NOT A

The contents of the accumulator are complemented (one's complement); all 1s are changed to 0 and vice-versa.

Flags:

S: Unaffected
Z: Unaffected
H: Set
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	CPL [A]	00101111	2	

CPLW
COMPLEMENT HL REGISTER (WORD)

CPLW [HL]

Operation: HL(15-0) ← NOT HL(15-0)

The contents of the HL register are complemented (ones complement); all 1s are changed to 0 and vice-versa.

Flags:

S: Unaffected
Z: Unaffected
H: Set
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	CPLW [HL]	11011101 00101111	2	

DAA
DECIMAL ADJUST ACCUMULATOR

DAA

Operation: A ← Decimal Adjust A

The accumulator is adjusted to form two 4-bit BCD digits following a binary, two's complement addition or subtraction on two BCD-encoded bytes. The table below indicates the operation performed for addition (ADD, ADC, INC) or subtraction (SUB, SBC, DEC, NEG).

Operation	C Before DAA	Hex Value Upper Digit (Bits 7-4)	H Before DAA	Hex Value Lower Digit (Bits 3-0)	Number Added to Byte	C After DAA	H After DAA
	0	0-9	0	0-9	00	0	0
	0	0-8	0	A-F	06	0	1
ADD	0	0-9	1	0-3	06	0	0
ADC	0	A-F	0	0-9	60	1	0
INC	0	9-F	0	A-F	66	1	1
(N=0)	0	A-F	1	0-3	66	1	0
	1	0-2	0	0-9	60	1	0
	1	0-2	0	A-F	66	1	1
	1	0-3	1	0-3	66	1	0
SUB							
SBC	0	0-9	0	0-9	00	0	0
DEC	0	0-8	1	6-F	FA	0	1
NEG	1	7-F	0	0-9	A0	1	0
(N=1)	1	6-F	1	6-F	9A	1	1

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: See table above
P: Set if the parity of the result is even; cleared otherwise
N: Not affected
C: See table above

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	DAA	00100111	3	

DDIR
DECODER DIRECTIVE

DDIR mode mode = W or LW, IB or IW

Operation: None, decoder directive only

This is not an instruction, but rather a directive to the instruction decoder.

The instruction decoder may be directed to fetch an additional byte or word of immediate data or address with the instruction, as well as tagging the instruction for execution in either Word or Long Word mode. All eight combinations of the two options are supported, as shown in the encoding below. Instructions which do not support decoder directives are assembled by the instruction decoder as if the decoder directive were not present.

The IB decoder directive causes the decoder to fetch an additional byte immediately after the existing immediate data or direct address, and in front of any trailing opcode bytes (with instructions starting with DD-CB or FD-CB, for example).

Likewise, the IW decoder directive causes the decoder to fetch an additional word immediately after the existing immediate data or direct address, and in front of any trailing opcode bytes.

Byte ordering within the instruction follows the usual convention; least significant byte first, followed by more significant bytes. More-significant immediate data or direct address bytes not specified in the instruction are taken as all zeros by the processor.

The W decoder directive causes the instruction decoder to tag the instruction for execution in Word mode. This is useful while the Long Word (LW) bit in the Select Register (SR) is set, but 16-bit data manipulation is required for this instruction.

The LW decoder directive causes the instruction decoder to tag the instruction for execution in Long Word mode. This is useful while the LW bit in the SR is cleared, but 32-bit data manipulation is required for this instruction.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	DDIR mode	11w11101 110000im	0	

Field Encodings:

wim:		
000	W	Word mode
001	IB,W	Immediate byte, Word mode
010	IW,W	Immediate word, Word mode
011	IB	Immediate byte
100	LW	Long Word mode
101	IB,LW	Immediate byte, Long Word mode
110	IW,LW	Immediate word, Long Word mode
111	IW	Immediate word

DEC
DECREMENT (BYTE)

DEC dst dst = R, RX, IR, X

Operation: dst ← dst – 1

The destination operand is decremented by one and the result is stored in the destination. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 4 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the destination was 80H; cleared otherwise
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	DEC R	00-r-101	note	
RX:	DEC RX	11y11101 0010w101	2	
IR:	DEC (HL)	00110101	2+r+w	
X:	DEC (XY+d)	11y11101 00110101 ——d—	4+r+w	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

Note: 2 for accumulator, 3 for any other register

DEC[W]
DECREMENT (WORD)

DEC[W] dst dst = R, RX

Operation:

```
if (XM) then begin
    dst(31-0)  ←  dst(31-0) - 1
    end
else begin
    dst(15-0)  ←  dst(15-0) - 1
    end
```

The destination operand is decremented by one and the result is stored in the destination. Two's complement subtraction is performed. Note that the length of the operand is controlled by the Extended/Native mode selection, which is consistent with the manipulation of an address by the instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	DEC[W] R	00rr1011	2	X
RX:	DEC[W] RX	11y11101 00101011	2	X

Field Encodings:
rr: 00 for BC, 01 for DE, 10 for HL, 11 for SP
y: 0 for IX, 1 for IY

DI
DISABLE INTERRUPTS

DI [n]

Operation:

```
if (n is present) then begin
    for i=1 to 4 begin
        if (n(i) = 1) then begin
            IER(i-1)    ←    0
            end
        end
    if (n(0) = 1) then begin
        SR(5)       ←    0
        end
    end
else begin
    SR(5)           ←    0
    end
```

If an argument is present, disable the selected interrupts by clearing the appropriate enable bits in the Interrupt Enable Register, and then clear the Interrupt Enable Flag (IEF1) in the Select Register (SR) if the least-significant bit of the argument is set, disabling maskable interrupts. Bits 7-5 of the argument are ignored.

If no argument is present, IEF1 in the SR is set to 0, disabling maskable interrupts.

Note that during execution of this instruction the maskable interrupts are not sampled.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	DI	11110011	2	
	DI n	11011101 11110011 —n——	2	

DIVUW [HL,]src src = R, RX, IM, X

Operation:

HL(15-0) ← HL / src
HL(31-16) ← remainder

The contents of the the HL register (dividend) are divided by the source operand (divisor) and the quotient is stored in the lower word of the HL register; the remainder is stored in the upper word of the HL register. The contents of the source are unaffected. Both operands are treated as unsigned, binary integers. There are three possible outcomes of the DIVUW instruction, depending on the division and the resulting quotient:

Case 1: If the quotient is less than 65536, then the quotient is left in the HL register, the Overflow and Sign flags are cleared to 0, and the Zero flag is set according to the value of the quotient.

Case 2: If the divisor is zero, the HL register is unchanged, the Zero and Overflow flags are set to 1, and the Sign flag is cleared to 0.

Case 3: If the quotient is greater than or equal to 65536, the HL register is unchanged, the Overflow flag is set to 1, and the Sign and Zero flags are cleared to 0.

Flags:

S: Cleared
Z: Set if the quotient or divisor is zero; cleared otherwise
H: Unaffected
V: Set if the divisor is zero or if the computed quotient is greater than or equal to 65536; cleared otherwise
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	DIVUW [HL,]R	11101101 11001011 101110rr	20	
RX:	DIVUW [HL,]RX	11101101 11001011 1011110y	20	
IM:	DIVUW [HL,]nn	11101101 11001011 10111111 -n(low)- -n(high)	20	
X:	DIVUW [HL,](XY+d)	11y11101 11001011 ——d— 10111010	22+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

DJNZ
DECREMENT AND JUMP IF NON-ZERO

DJNZ dst dst = RA

Operation:

```
B                    ←   B-1
If (B <> 0) then begin
    dst              ←   SIGN EXTEND dst
    if (XM) then begin
        PC(31-0)     ←   PC(31-0) + dst(31-0)
        end
    else begin
    PC(15-0)         ←   PC(15-0) + dst(15-0)
    end
end
```

The B register is decremented by one. If the result is non-zero, then the destination address is calculated and then loaded into the Program Counter (PC). Control then passes to the instruction whose address is pointed to by the PC. When the B register reaches zero, control falls through to the instruction following DJNZ. This instruction provides a simple method of loop control.

The destination address is calculated using Relative addressing. The displacement in the instruction is added to the PC; the PC value used is the address of the instruction following the DJNZ instruction.

These instructions employ either an 8-bit, 16-bit, or 24-bit signed, two's complement displacement from the PC to permit jumps within a range of -126 to +129 bytes, -32,765 to +32,770 bytes, or -8,388,604 to +8,388,611 bytes from the location of this instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
RA:	DJNZ addr	00010000 —disp—	note	X
	DJNZ addr	11011101 00010000 -d(low)- -d(high)	note	X
	DJNZ addr	11111101 00010000 -d(low)- -d(mid)- -d(high)	note	X

Note: 3 if branch not taken, 4 if branch taken

EI
ENABLE INTERRUPTS

EI [n]

Operation:

```
if (n is present) then begin
    for i=1 to 4 begin
        if (n(i) = 1) then begin
            IER(i-1)    ←    1
            end
        end
    if (n(0) = 1) then begin
        SR(5)           ←    1
        end
    end
else begin
    SR(5)               ←    1
    end
```

If an argument is present, enable the selected interrupts by setting the appropriate enable bits in the Interrupt Enable Register, and then set the Interrupt Enable Flag (IEF1) in the Select Register (SR) if the least-significant bit of the argument is set, enabling maskable interrupts. Bits 7-5 of the argument are ignored.

If no argument is present, IEF1 in the SR is set to 1, enabling maskable interrupts.

Note that during the execution of this instruction and the following instruction, maskable interrupts are not sampled.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	EI	11111011	2	
	EI n	11011101 11111011 —n——	2	

EX
EXCHANGE ACCUMULATOR/FLAG WITH ALTERNATE BANK

EX AF,AF'

Operation: SR(0) ← NOT SR(0)

Bit 0 of the Select Register (SR), which controls the selection of primary or alternate bank for the accumulator and flag register, is complemented, thus effectively exchanging the accumulator and flag registers between the two banks.

Flags:

S: Value in F'
Z: Value in F'
H: Value in F'
V: Value in F'
N: Value in F'
C: Value in F'

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	EX AF,AF'	00001000	3	

EX
EXCHANGE ADDRESSING REGISTER WITH TOP OF STACK

EX (SP),dst   dst = HL, IX, IY

Operation:

```
if (LW) then begin
    (SP+3) ↔ dst(31-24)
    (SP+2) ↔ dst(23-16)
    end
(SP+1)     ↔ dst(15-8)
(SP)       ↔ dst(7-0)
```

The contents of the destination register are exchanged with the top of the stack. In Long Word mode this exchange is two words; otherwise it is one word.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	EX (SP),HL	11100011	3+r+w	L
	EX (SP),XY	11y11101 11100011	3+r+w	L

Field Encodings: y: 0 for IX, 1 for IY

EX
EXCHANGE REGISTER (WORD)

EX dst,src   dst = R, RX
src = R, RX

Operation:

```
if (LW) then begin
    dst(31-0)  ↔  src(31-0)
    end
else begin
    dst(15-0)  ↔  src(15-0)
    end
```

The contents of the destination are exchanged with the contents of the source.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	EX BC,DE	11101101 00000101	3	L
	EX BC,HL	11101101 00001101	3	L
	EX DE,HL	11101011	3	L
RX:	EX R,RX	11101101 00rry011	3	L
	EX IX,IY	11101101 00101011	3	L

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

EX
EXCHANGE REGISTER WITH ALTERNATE REGISTER (BYTE)

EX dst,src   src = R

Operation: dst ↔ src

The contents of the destination are exchanged with the contents of the source, where the destination is a register in the primary bank and the source is the corresponding register in the alternate bank

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	EX R,R'	11001011 00110-r-	3	

Field Encoding: r: per convention

EX
EXCHANGE REGISTER WITH ALTERNATE REGISTER (WORD)

EX dst,src src = R, RX

Operation:

```
if (LW) then begin
    dst(31-0)  ↔  src(31-0)
    end
else begin
    dst(15-0)  ↔  src(15-0)
    end
```

The contents of the destination are exchanged with the contents of the source, where the destination is a word register in the primary bank and the source is the corresponding word register in the alternate bank.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	EX R,R'	11101101 11001011 001100rr	3	L
RX:	EX RX,RX'	11101101 11001011 0011010y	3	L

Field Encodings:
rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

EX
EXCHANGE WITH ACCUMULATOR

EX A,src src = R, IR

Operation: dst ↔ src

The contents of the accumulator are exchanged with the contents of the source.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	EX A,R	11101101 00-r-111	3	
IR:	EX A,(HL)	11101101 00110111	3+r+w	

Field Encodings: r: per convention

EXALL
EXCHANGE ALL REGISTERS WITH ALTERNATE BANK

EXALL

Operation:

SR(24) ← NOT SR(24)
SR(16) ← NOT SR(16)
SR(8) ← NOT SR(8)

Bits 8, 16, and 24 of the Select Register (SR), which control the selection of primary or alternate bank for the BC, DE, HL, IX, and IY registers, are complemented, thus effectively exchanging the BC, DE, HL, IX, and IY registers between the two banks.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Unaffected
V:	Unaffected
N:	Unaffected
C:	Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	EXALL	11101101 11011001	3	

EXTS
EXTEND SIGN (BYTE)

EXTS [A]

Operation:

```
L                       ←      A
if (A(7)=0) then begin
    H   ¨   00h
    if (LW) then begin
        HL(31-16)       ←      0000h
        end
    end
else begin
    H   ¨   FFh
    if (LW) then begin
        HL(31-16)       ←      FFFFh
        end
    end
```

The contents of the accumulator, considered as a signed, two's complement integer, are sign-extended to 16 bits and the result is stored in the HL register. The contents of the accumulator are unaffected. This instruction is useful for conversion of short signed operands into longer signed operands.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	EXTS [A]	11101101 01100101	3	L

EXTSW
EXTEND SIGN (WORD)

EXTSW [HL]

Operation:

```
If (HL(15)=0) then begin
    HL(31-16)  <-   0000h
    end
else begin
    HL(31-16)  <-   FFFFh
    end
```

The contents of the low word of the HL register, considered as a signed, two's complement integer, are sign-extended to 32 bits in the HL register. This instruction is useful for conversion of 16-bit signed operands into 32-bit signed operands.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	EXTSW [HL]	11101101 01110101	3	

EXX
EXCHANGE REGISTERS WITH ALTERNATE BANK

EXX

Operation: SR(8) ← NOT SR(8)

Bit 8 of the Select Register (SR), which controls the selection of primary or alternate bank for the BC, DE, and HL registers, is complemented, thus effectively exchanging the BC, DE, and HL registers between the two banks.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	EXX	11011001	3	

EXXX
EXCHANGE IX REGISTER WITH ALTERNATE BANK

EXXX

Operation: SR(16) ← NOT SR(16)

Bit 16 of the Select Register (SR), which controls the selection of primary or alternate bank for the IX register, is complemented, thus effectively exchanging the IX register between the two banks.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	EXXX	11011101 11011001	3	

EXXY
EXCHANGE IY REGISTER WITH ALTERNATE BANK

EXXY

Operation: SR(24) ← NOT SR(24)

Bit 24 of the Select Register (SR), which controls the selection of primary or alternate bank for the IY register, is complemented, thus effectively exchanging the IY register between the two banks.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	EXXY	11111101 11011001	3	

HALT
HALT

HALT

Operation: CPU Halts

The CPU operation is suspended until either an interrupt request or reset request is received. This instruction is used to synchronize the CPU with external events, preserving its state until an interrupt or reset request is accepted. After an interrupt is serviced, the instruction following HALT is executed. While the CPU is halted, memory refresh cycles still occur, and bus requests are honored. When this instruction is executed the signal /HALT is asserted and remains asserted until an interrupt or reset request is accepted.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	HALT	01110110	2	

IM
INTERRUPT MODE SELECT

IM p   p = 0, 1, 2, 3

Operation: SR(4-3) ← p

The interrupt mode of operation is set to one of four modes. (See Chapter 6 for a description of the various modes for responding to interrupts). The current interrupt mode can be read from the Select Register (SR).

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	IM p	11101101 010pp110	4	

Field Encodings: pp: 00 for Mode 0, 01 for Mode 3, 10 for Mode 1, 11 for Mode 2

IN
INPUT (BYTE)

IN dst,(C) dst = R

Operation: dst ← (C)

The byte of data from the selected peripheral is loaded into the destination register. During the I/O transaction, the contents of the 32-bit BC register are placed on the address bus.

Flags:

S:	Set if the input data is negative; cleared otherwise
Z:	Set if the input data is zero; cleared otherwise
H:	Cleared
P:	Set if the input data has even parity; cleared otherwise
N:	Cleared
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	IN R,(C)	11101101 01-r-000	2+i	

Field Encodings: r: per convention

INW
INPUT (WORD)

INW dst,(C) dst = R

Operation: dst(15-0) ← (C)

The word of data from the selected peripheral is loaded into the destination register. During the I/O transaction, the contents of the 32-bit BC register are placed on the address bus.

Flags:

S: Set if the input data is negative; cleared otherwise
Z: Set if the input data is zero; cleared otherwise
H: Cleared
P: Set if the input data has even parity; cleared otherwise
N: Cleared
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	INW R,(C)	11011101 01rrr000	2+i	

Field Encodings: rrr: 000 for BC, 010 for DE, 111 for HL

IN
INPUT ACCUMULATOR

IN A,(n)

Operation: A ← (n)

The byte of data from the selected peripheral is loaded into the accumulator. During the I/O transaction, the 8-bit peripheral address from the instruction is placed on the low byte of the address bus, the contents of the accumulator are placed on address lines A15-A8, and the high-order address lines are all zeros.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	IN A,(n)	11011011 ——n—	3+i	

INO
INPUT (FROM PAGE 0)

INO dst,(n) dst = R

Operation: dst ← (n)

The byte of data from the selected on-chip peripheral is loaded into the destination register. No external I/O transaction will be generated as a result of this instruction, although the I/O address will appear on the address bus while this internal read is occurring. The peripheral address is placed on the low byte of the address bus and zeros are placed on all other address lines. When the second opcode byte is 30h no data is stored in a destination; only the flags are updated.

Flags:

S:	Set if the input data is negative; cleared otherwise
Z:	Set if the input data is zero; cleared otherwise
H:	Cleared
P:	Set if the input data has even parity; cleared otherwise
N:	Cleared
C:	Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	INO R,(n)	11101101 00 -r- 000 ——n—	3+i	
none:	INO (n	11101101 00110000 ——n—	3+i	

Field Encodings: r: per convention

INA
INPUT DIRECT FROM PORT ADDRESS (BYTE)

INA A,(nn)

Operation: A ← (nn)

The byte of data from the selected peripheral is loaded into the accumulator. During the I/O transaction, the peripheral address from the instruction is placed on the address bus. Any bytes of address not specified in the instruction are driven on the address lines as all zeros.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INA A,(nn)	11101101 11011011 -n(low)- -n(high)	3+i	I

INAW
INPUT DIRECT FROM PORT ADDRESS (WORD)

INAW HL,(nn)

Operation: HL(15-0) ← (nn)

The word of data from the selected peripheral is loaded into the HL register. During the I/O transaction, the peripheral address from the instruction is placed on the address bus. Any bytes of address not specified in the instruction are driven on the address lines as all zeros.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INAW HL,(nn)	11111101 11011011 -n(low)- -n(high)	3+i	I

INC
INCREMENT (BYTE)

INC dst dst = R, RX, IR, X

Operation: dst ← dst + 1

The destination operand is incremented by one and the sum is stored in the destination. Two's complement addition is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a carry from bit 3 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the destination was 7FH; cleared otherwise
N: Cleared
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	INC R	00-r-100	note	
RX:	INC RX	11y11101 0010w100	2	
IR:	INC (HL)	00110100	2+r+w	
X:	INC (XY+d)	11y11101 00110100 ——d—	4+r+w	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

Note: 2 for accumulator, 3 for any other register

INC[W]
INCREMENT (WORD)

INC[W] dst dst = R, RX

Operation:

```
if (XM) then begin
    dst(31-0)    <     dst(31-0) + 1
    end
else begin
    dst(15-0)    ←     dst(15-0) + 1
    end
```

The destination operand is incremented by one and the sum is stored in the destination. Two's complement addition is performed. Note that the length of the operand is controlled by the Extended/Native mode selection, which is consistent with the manipulation of an address by the instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	INC[W] R	00rr0011	2	X
RX:	INC[W] RX	11y11101 00100011	2	X

Field Encodings: rr: 00 for BC, 01 for DE, 10 for HL, 11 for SP
y: 0 for IX, 1 for IY

IND
INPUT AND DECREMENT (BYTE)

IND

Operation:

(HL) ← (C)
B ← B – 1
HL ← HL – 1

This instruction is used for block input of strings of data. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A15-A8 are not useable as part of a fixed port address.

First the byte of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the B register, used as a counter, is decremented by one. The HL register is then decremented by one, thus moving the pointer to the next destination for the input.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	IND	11101101 10101010	2+i+w	

INDW
INPUT AND DECREMENT (WORD)

INDW

Operation:

(HL) ← (DE)
BC(15-0) ← BC(15-0) – 1
HL ← HL – 2

This instruction is used for block input of strings of data. During the I/O transaction the 32-bit DE register is placed on the address bus.

First the word of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the BC register, used as a counter, is decremented by one. The HL register is then decremented by two, thus moving the pointer to the next destination for the input.

Flags:

S: Unaffected
Z: Set if the result of decrementing BC is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INDW	11101101 11101010	2+i+w	

INDR
INPUT, DECREMENT AND REPEAT (BYTE)

INDR

Operation:

```
repeat until (B=0) begin
    (HL)    ←   (C)
    B       ←   B - 1
    HL      ←   HL - 1
    end
```

This instruction is used for block input of strings of data. The string of input data from the selected peripheral is loaded into memory at consecutive addresses, starting with the location addressed by the HL register and decreasing. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A15-A8 are not useable as part of a fixedport address.

First the byte of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the B register, used as a counter, is decremented by one. The HL register is then decremented by one, thus moving the pointer to the next destination for the input. If the result of decrementing the B register is 0, the instruction is terminated, otherwise the sequence is repeated. If the B register contains 0 at the start of the execution of this instruction, 256 bytes are input.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INDR	11101101 10111010	n X (2+i+w)	

INDRW
INPUT, DECREMENT AND REPEAT (WORD)

INDRW

Operation:

```
repeat until (BC=0) begin
    (HL)        ←    (DE)
    BC(15-0)    ←    BC(15-0) – 1
    HL          ←    HL – 2
    end
```

This instruction is used for block input of strings of data. The string of input data from the selected peripheral is loaded into memory at consecutive addresses, starting with the location addressed by the HL register and decreasing. During the I/O transaction the 32-bit DE register is placed on the address bus.

First the BC register, used as a counter, is decremented by one. First the word of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the BC register, used as a counter, is decremented by one. The HL register is then decremented by two, thus moving the pointer to the next destination for the input. If the result of decrementing the BC register is 0, the instruction is terminated, otherwise the sequence is repeated. If the BC register contains 0 at the start of the execution of this instruction, 65536 bytes are input.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing BC is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	INDRW	11101101 11111010	n X (2+i+w)	

INI
INPUT AND INCREMENT (BYTE)

INI

Operation:

(HL) ← (C)
B ← B – 1
HL ← HL + 1

This instruction is used for block input of strings of data. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A15-A8 are not useable as part of a fixed port address.

First the byte of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the B register, used as a counter, is decremented by one. The HL register is then incremented by one, thus moving the pointer to the next destination for the input.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INI	11101101 10100010	2+i+w	

INIW
INPUT AND INCREMENT (WORD)

INIW

Operation:

(HL)	←	(DE)
BC(15-0)	←	BC(15-0) – 1
HL	←	HL + 2

This instruction is used for block input of strings of data.
During the I/O transaction the 32-bit DE register is placed on the address bus.

First the word of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the BC register, used as a counter, is decremented by one. The HL register is then incremented by two, thus moving the pointer to the next destination for the input.

Flags:

S: Unaffected
Z: Set if the result of decrementing BC is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INIW	11101101 11100010	2+i+w	

INIR
INPUT, INCREMENT AND REPEAT (BYTE)

INIR

Operation:

```
repeat until (B=0) begin
    (HL)  ←  (C)
    B     ←  B – 1
    HL    ←  HL + 1
    end
```

This instruction is used for block input of strings of data. The string of input data from the selected peripheral is loaded into memory at consecutive addresses, starting with the location addressed by the HL register and increasing. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A(15-8) are not useable as part of a fixedport address.

First the byte of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the B register, used as a counter, is decremented by one. The HL register is then incremented by one, thus moving the pointer to the next destination for the input. If the result of decrementing the B register is 0, the instruction is terminated, otherwise the sequence is repeated. If the B register contains 0 at the start of the execution of this instruction, 256 bytes are input.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INIR	11101101 10110010	n X (2+i+w)	

INIRW
INPUT, INCREMENT AND REPEAT (WORD)

INIRW

Operation:

```
repeat until (BC=0) begin
    (HL)        ←    (DE)
    BC(15-0)    ←    BC(15-0) – 1
    HL          ←    HL + 2
    end
```

This instruction is used for block input of strings of data. The string of input data from the selected peripheral is loaded into memory at consecutive addresses, starting with the location addressed by the HL register and increasing. During the I/O transaction the 32-bit DE register is placed on the address bus.

First the word of data from the selected peripheral is loaded into the memory location addressed by the HL register. Then the BC register, used as a counter, is decremented by one. The HL register is then incremented by two, thus moving the pointer to the next destination for the input. If the result of decrementing the BC register is 0, the instruction is terminated, otherwise the sequence is repeated. If the BC register contains 0 at the start of the execution of this instruction, 65536 bytes are input.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing BC is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	INIRW	11101101 11110010	n X (2+i+w)	

JP
JUMP

JP [cc,]dst dst = IR, DA

Operation:

```
if (cc is TRUE) then begin
    if (XM) then begin
        PC(31-0)    ←    dst(31-0)
        end
    else begin
        PC(15-0)    ←    dst(15-0)
        end
    end
```

A conditional jump transfers program control to the destination address if the setting of a selected flag satisfies the condition code "cc" specified in the instruction; an unconditional jump always transfers control to the destination address. If the jump is taken, the Program Counter (PC) is loaded with the destination address; otherwise the instruction following the Jump instruction is executed.

Each of the Zero, Carry, Sign, and Overflow flags can be individually tested and a jump performed conditionally on the setting of the flag.

When using DA mode with the JP instruction, the operand is not enclosed in parentheses.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
IR:	JP (HL)	11101001	2	X
	JP (XY)	11y11101 11101001	2	X
DA:	JP CC,addr	11-cc010 -a(low)- -a(high)	2	I, X
	JP addr	11000011 -a(low)- -a(high)	2	I, X

Field Encodings: y: 0 for IX, 1 for IY

cc: 000 for NZ, 001 for Z, 010 for NC, 011 for C, 100 for PO/NV, 101 for PE/V, 110 for P/NS,111 for M/S

JR
JUMP RELATIVE

JR [cc,]dst dst = RA

Operation:

```
if (cc is TRUE) then begin
    dst ←  SIGN EXTEND dst
    if (XM) then begin
        PC(31-0)     <-     PC(31-0) + dst(31-0)
        end
    else begin
        PC(15-0)     <-     PC(15-0) + dst(15-0)
        end
    end
```

A conditional Jump transfers program control to the destination address if the setting of a selected flag satisfies the condition code "cc" specified in the instruction; an unconditional Jump always transfers control to the destination address. Either the Zero or Carry flag can be tested for the conditional Jump. If the jump is taken, the Program Counter (PC) is loaded with the destination address; otherwise the instruction following the Jump Relative instruction is executed.

The destination address is calculated using relative addressing. The displacement in the instruction is added to the PC value for the instruction following the JR instruction, not the value of the PC for the JR instruction.

These instructions employ either an 8-bit, 16-bit, or 24-bit signed, two's complement displacement from the PC to permit jumps within a range of –126 to +129 bytes, –32,765 to +32,770 bytes, or –8,388,604 to +8,388,611 bytes from the location of this instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
RA:	JR CC,addr	001cc000 —disp—	2	X
	JR addr	00011000 —disp—	2	X
	JR CC,addr	11011101 001cc000 -d(low)- -d(high)	2	X
	JR addr	11011101 00011000 -d(low)- -d(high)	2	X
	JR CC,addr	11111101 001cc000 -d(low)- -d(mid)- -d(high)	2	X
	JR addr	11111101 00011000 -d(low)- -d(mid)- -d(high)	2	X

Field Encodings: cc: 00 for NZ, 01 for Z, 10 for NC, 11 for C

LD
LOAD ACCUMULATOR

LD dst,src — dst = A
src = R, RX, IM, IR, DA, X
or
dst = R, RX, IR, DA, X
src = A

Operation: dst ← src

The contents of the source are loaded into the destination.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load into Accunulator

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD A,R	01111-r-	2	
RX:	LD A,RX	11y11101 0111110w	2	
IM:	LD A,n	00111110 ——n—	2	
IR:	LD A,(HL)	01111110	2+r	
	LD A,(IR)	000a1010	2+r	
DA:	LD A,(nn)	00111010 -n(low)- -n(high)	3+r	I
X:	LD A,(XY+d)	11y11101 01111110 ——d—	4+r	I

Load from Accumulator

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD Rd,A	01-r-111	2	
RX:	LD RX,A	11y11101 0110w111	2	
IR:	LD (HL),A	01110111	3+w	
	LD (IR),A	000a0010	3+w	
DA:	LD (nn),A	00110010 -n(low)- -n(high)	4+w	I
X:	LD (XY+d),A	11y11101 01110111 ——d—	5+w	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte
a: 0 for BC, 1 for DE

LD
LOAD IMMEDIATE (BYTE)

LD dst,n   dst = R, RX, IR, X

Operation: dst ← n

The byte of immediate data is loaded into the destination.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD R,n	00-r-110 ——n—	2	
RX:	LD RX,n	11y11101 0010w110 ——n—	2	
IR:	LD (HL),n	00110110 ——n—	3+w	
X:	LD (XY+d),n	11y11101 00110110 ——d— ——n—	5+w	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

LD
LOAD IMMEDIATE (WORD)

LD dst,nn dst = R, RX

Operation:

```
if (LW) then begin
    dst(31-0)  <-   nn
    end
else begin
    dst(15-0)  <-   nn
    end
```

The word of immediate data is loaded into the destination.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD R,nn	00rr0001 -n(low)- -n(high)	2	I, L
RX:	LD RX,nn	11y11101 00100001 -n(low)- -n(high)	2	I, L

Field Encodings: rr: 00 for BC, 01 for DE, 10 for HL
y: 0 for IX, 1 for IY

LDW
LOAD IMMEDIATE (WORD)

LDW dst,nn   dst = IR

Operation:

```
if (LW) then begin
    dst(31-0)   ←   nn
    end
else begin
    dst(15-0)   ←   nn
end
```

The word of immediate data is loaded into the destination.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
IR:	LDW (IR),nn	11101101 00pp0110 -n(low)- -n(high)	3+w	I, L

Field Encodings: pp: 00 for BC, 01 for DE, 11 for HL

LD
LOAD REGISTER (BYTE)

LD dst,src dst = R
src = R, RX, IM, IR, X
or
dst = R, RX, IR, X
src = R

Operation: dst ← src

The contents of the source are loaded into the destination.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load into Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD Rd,Rs	01-rd-rs	2	
RX:	LD Rd,RX	11y11101 01-ra10w	2	
	LD RXa,RXb	11y11101 0110a10b	2	
IM:	LD R,n	00-r-110 ——n—	2	
IR:	LD R,(HL)	01-r-110	5+w	
X:	LD R,(XY+d)	11y11101 01-r-110 ——d—	7+w	I

Load from Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
RX:	LD RX,Rs	11y11101 0110w-ra	2	
	LD RXa,RXb	11y11101 0110a10b	2	
IR:	LD (HL),R	01110-r-	3+w	
X:	LD (XY+d),R	11y11101 01110-r- ——d—	5+w	I

Field Encodings:
r: per convention
rd: per convention
rs: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte
ra: per convention, for A, B, C, D, E only
a: destination, 0 for high byte, 1 for low byte
b: source, 0 for high byte, 1 for low byte

LD[W]
LOAD REGISTER (WORD)

LD[W] dst,src dst = R
src = R, RX, IR, DA, X, SR
or
dst = R, RX, IR, DA, X, SR
src = R

Operation:

```
if (LW) then begin
    dst(31-0)  ←  src(31-0)
    end
else begin
    dst(15-0)  ←  src(15-0)
    end
```

The contents of the source are loaded into the destination.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load into Register Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD Rd,Rs	11rs1101 00rd0010	2	L
RX:	LD R,RX	11y11101 00rr1011	2	L
IR:	LD R,(IR)	11011101 00rr11ri	2+r	L
	LD RX,(IR)	11y11101 00ri0011	2+r	L
DA:	LD HL,(nn)	00101010 -n(low)- -n(high)	3+r	I, L
	LD R,(nn)	11101101 01ra1011 -n(low)- -n(high)	3+r	I, L
	LD RX,(nn)	11y11101 00101010 -n(low)- -n(high)	3+r	I, L
X:	LD R,(XY+d)	11y11101 11001011 ——d— 00rr0011	4+r	I, L
	LD IX,(IY+d)	11111101 11001011 ——d— 00100011	4+r	I, L
	LD IY,(IX+d)	11011101 11001011 ——d— 00100011	4+r	I, L
SR:	LD R,(SP+d)	11011101 11001011 ——d— 00rr0001	4+r	I, L
	LD RX,(SP+d)	11y11101 11001011 ——d— 00100001	4+r	I, L

LD[W]
LOAD REGISTER (WORD)

Load from Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
RX:	LD RX,R	11y11101 00rr0111	2	L
	LD IX,IY	11011101 00100111	2	L
	LD IY,IX	11111101 00100111	2	L
IR:	LD (IR),RR	11111101 00rr11ri	3+w	L
	LD (IR),RX	11y11101 00ri0001	3+w	L
DA:	LD (nn),HL	00100010 -n(low)- -n(high)	4+w	I, L
	LD (nn),R	11101101 01ra0011 -n(low)- -n(high)	4+w	I, L
	LD (nn),RX	11y11101 00100010 -n(low)- -n(high)	4+w	I, L
X:	LD (XY+d),R	11y11101 11001011 ——d— 00rr1011	5+w	I, L
	LD (IY+d),IX	11111101 11001011 ——d— 00101011	5+w	I, L
	LD (IX+d),IY	11011101 11001011 ——d— 00101011	5+w	I, L
SR:	LD (SP+d),R	11011101 11001011 ——d— 00rr1001	5+w	I, L
	LD (SP+d),XY	11y11101 11001011 ——d— 00101001	5+w	I, L

Field Encodings:
rs: 01 for DE, 10 for BC, 11 for HL
rd: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY
rr: 00 for BC, 01 for DE, 11 for HL
ri: 00 for BC, 01 for DE, 11 for HL
ra: 00 for BC, 01 for DE, 10 for HL

LD
LOAD STACK POINTER

LD dst,src   dst = SP
src = R, RX, IM, DA
or
dst = DA
src = SP

Operation:

```
if (LW) then begin
    dst(31-0)  ←  src(31-0)
    end
else begin
    dst(15-0)  ←  src(15-0)
    end
```

The contents of the source are loaded into the destination.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load into Stack Pointer

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD SP,HL	11111001	2	L
RX:	LD SP,RX	11y11101 11111001	2	L
IM:	LD SP,nn	00110001 -n(low)- -n(high)	2	I, L
DA:	LD SP,(nn)	11101101 01111011 -n(low)- -n(high)	3+r	I, L

Field Encodings: y: 0 for IX, 1 for IY

Load from Stack Pointer

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
DA:	LD (nn),SP	11101101 01110011 -n(low)- -n(high)	4+w	I, L

LD
LOAD FROM I OR R REGISTER (BYTE)

LD dst,src   dst = A
src = I, R

Operation: dst ← src

The contents of the source are loaded into the accumulator. The contents of the source are not affected. The Sign and Zero flags are set according to the value of the data transferred; the Overflow flag is set according to the state of the interrupt enable. Note that if an interrupt occurs during execution of either of these instructions the Overflow flag reflects the prior state of the interrupt enable. Also note that the R register does not contain the refresh address and is not modified by refresh transactions.

Flags:

S: Set if the data loaded into the accumulator is negative; cleared otherwise
Z: Set if the data loaded into the accumulator is zero; cleared otherwise
H: Cleared
V: Set when loading the accumulator if interrupts are enabled; cleared otherwise
N: Cleared
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	LD A,I	11101101 01010111	2	
	LD A,R	11101101 01011111	2	

LD
LOAD INTO I OR R REGISTER (BYTE)

LD dst,src dst = I, R
src = A

Operation: dst ← src

The contents of the accumulator are loaded into the destination. Note that the R register does not contain the refresh address and is not modified by refresh transactions.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Unaffected
V:	Unaffected
N:	Unaffected
C:	Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	LD I,A	11101101 01000111	2	
	LD R,A	11101101 01001111	2	

LD[W]
LOAD I REGISTER (WORD)

LD[W] dst,src

dst = HL
src = I
OR
dst = I
src = HL

Operation:

```
if (LW) then begin
    dst(31-0)  ←  src(31-0)
    end
else begin
    dst(15-0)  ←  src(15-0)
    end
```

The contents of the source are loaded into the destination

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load from I Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD[W] HL,I	11011101 01010111	2	L

Load into I Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LD[W] I,HL	11011101 01000111	2	L

LDCTL
LOAD CONTROL REGISTER (BYTE)

LDCTL dst,src

dst = DSR, XSR, YSR
src = A, IM
or
dst = A
src = DSR, XSR, YSR
or
dst = SR
src = A, IM

Operation:

```
if (dst = SR) then begin
    SR(31-24)  ←  src
    SR(23-16)  ←  src
    SR(15-8)   ←  src
    end
else begin
    dst        ←  src
    end
```

The contents of the source are loaded into the destination.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load into Control Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LDCTL SR,A	11011101 11001000	4	
	LDCTL Rd,A	11qq1101 11011000	4	
IM:	LDCTL SR,n	11011101 11001010 ——n—	4	
	LDCTL Rd,n	11qq1101 11011010 ——n—	4	

Field Encodings: qq: 01 for XSR, 10 for DSR, 11 for YSR

Load from Control Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LDCTL A,Rs	11qq1101 11010000	2	

Field Encodings: qq: 01 for XSR, 10 for DSR, 11 for YSR

LDCTL
LOAD FROM CONTROL REGISTER (WORD)

LDCTL dst,src   dst = HL
src = SR

Operation:

```
if (LW) then begin
    dst(31-0)  ←   src(31-0)
    end
else begin
    dst(15-0)  ←   src(15-0)
    end
```

The contents of the Select Register (SR) are loaded into the HL register.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load from Control Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LDCTL HL,SR	11101101 11000000	2	L

LDCTL
LOAD INTO CONTROL REGISTER (WORD)

LDCTL dst,src   dst = SR
                     src = HL

Operation:

```
if (LW) then begin
    dst(31-16)  ←   HL(31-16)
    end
else begin
    dst(31-24)  ←   HL(15-8)
    dst(23-16)  ←   HL(15-8)
    end
dst(15-8)       ←   HL(15-8)
dst(0)          ←   HL(0)
```

The contents of the HL register are loaded into the Select Register (SR). If Long Word mode is not in effect the upper byte of the HL register is copied into the three most significant bytes of the select register. This instruction does not modify the mode bits in the SR. There are dedicated instructions to modify the mode bits.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Load from Control Register

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	LDCTL SR,HL	11101101 11001000	4	L

LDD
LOAD AND DECREMENT (BYTE)

LDD

Operation:

(DE)	←	(HL)
DE	←	DE – 1
HL	←	HL – 1
BC(15-0)	←	BC(15-0) – 1

This instruction is used for block transfers of strings of data. The byte of data at the location addressed by the HL register is loaded into the location addressed by the DE register. Both the DE and HL registers are then decremented by one, thus moving the pointers to the preceeding elements in the string. The BC register, used as a counter, is then decremented by one.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Cleared
V:	Set if the result of decrementing BC is not equal to zero; cleared otherwise
N:	Cleared
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	LDD	11101101 10101000	3+r+w	

LDDW
LOAD AND DECREMENT (WORD)

LDDW

Operation:

```
if (LW) then begin
    (DE)        ←    (HL)
    (DE+1)      ←    (HL+1)
    (DE+2)      ←    (HL+2)
    (DE+3)      ←    (HL+3)
    DE          ←    DE – 4
    HL          ←    HL – 4
    BC(15-0)    ←    BC(15-0) – 4
    end
else begin
    (DE)        ←    (HL)
    (DE+1)      ←    (HL+1)
    DE          ←    DE – 2
    HL          ←    HL – 2
    BC(15-0)    ←    BC(15-0) – 2
    end
```

This instruction is used for block transfers of words of data. The word of data at the location addressed by the HL register is loaded into the location addressed by the DE register. Both the DE and HL registers are then decremented by two or four, thus moving the pointers to the preceeding words in the array. The BC register, used as a byte counter, is then decremented by two or four.

Both DE and HL should be even, to allow word transfers on the bus. BC must be even, transferring an even number of bytes, or the operation is undefined.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
V: Set if the result of decrementing BC is not equal to zero; cleared otherwise
N: Cleared
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	LDDW	11101101 11101000	3+r+w	L

LDDR
LOAD, DECREMENT AND REPEAT (BYTE)

LDDR

Operation:

```
repeat until BC=0 begin
    (DE)        <-    (HL)
    DE          <-    DE - 1
    HL          <-    HL - 1
    BC(15-0)    <-    BC(15-0) - 1
    end
```

This instruction is used for block transfers of strings of data. The bytes of data at the location addressed by the HL register are loaded into memory starting at the location addressed by the DE register. The number of bytes moved is determined by the contents of the BC register. If the BC register contains zero when this instruction is executed, 65,536 bytes are transferred. The effect of decrementing the pointers during the transfer is important if the source and destination strings overlap with the source string starting at a lower memory address. Placing the pointers at the highest address of the strings and decrementing the pointers ensures that the source string is copied without destroying the overlapping area.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value of the start of this instruction is saved before the interrupt request is accepted,so that the instruction can be properly resumed.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Cleared
V:	Cleared
N:	Cleared
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	LDDR	11101101 10111000	n X (3+r+w)	

LDDRW
LOAD, DECREMENT AND REPEAT (WORD)

LDDRW

Operation:

```
repeat until (BC=0) begin
    if (LW) then begin
        (DE)        ←   (HL)
        (DE+1)      ←   (HL+1)
        (DE+2)      ←   (HL+2)
        (DE+3)      ←   (HL+3)
        DE          ←   DE – 4
        HL          ←   HL – 4
        BC(15-0)    ←   BC(15-0) – 4
        end
    else begin
        (DE)        ←   (HL)
        (DE+1)      ←   (HL+1)
        DE          ←   DE – 2
        HL          ←   HL – 2
        BC(15-0)    ←   BC(15-0) – 2
        end
    end
```

This instruction is used for block transfers of strings of data. The words of data at the location addressed by the HL register are loaded into memory starting at the location addressed by the DE register. The number of words moved is determined by the contents of the BC register. If the BC register contains zero when this instruction is executed, 65,536 words are transferred. The effect of decrementing the pointers during the transfer is important if the source and destination strings overlap with the source string starting at a lower memory address. Placing the pointers at the highest address of the strings and decrementing the pointers ensures that the source string is copied without destroying the overlapping area.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value of the start of this instruction is saved before the interrupt request is accepted,so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
V: Cleared
N: Cleared
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	LDDRW	11101101 11111000	nX(3+r+w)	L

LDI
LOAD AND INCREMENT (BYTE)

LDI

Operation:

(DE) ← (HL)
DE ← DE + 1
HL ← HL + 1
BC(15-0) ← BC(15-0) – 1

This instruction is used for block transfers of strings of data. The byte of data at the location addressed by the HL register is loaded into the location addressed by the DE register. Both the DE and HL registers are then incremented by one, thus moving the pointers to the next elements in the string. The BC register, used as a counter, is then decremented by one.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
V: Set if the result of decrementing BC is not equal to zero; cleared otherwise
N: Cleared
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	LDI	11101101 10100000	3+r+w	

LDIW
LOAD AND INCREMENT (WORD)

LDIW

Operation:

```
if (LW) then begin
    (DE)        ←    (HL)
    (DE+1)      ←    (HL+1)
    (DE+2)      ←    (HL+2)
    (DE+3)      ←    (HL+3)
    DE          ←    DE + 4
    HL          ←    HL + 4
    BC(15-0)    ←    BC(15-0) – 4
    end
else begin
    (DE)        ←    (HL)
    (DE+1)      ←    (HL+1)
    DE          ←    DE + 2
    HL          ←    HL + 2
    BC(15-0)    ←    BC(15-0) – 2
    end
```

This instruction is used for block transfers of words of data. The word of data at the location addressed by the HL register is loaded into the location addressed by the DE register. Both the DE and HL registers are then incremented by two or four, thus moving the pointers to the succeeding words in the array. The BC register, used as a byte counter, is then decremented by two or four.

Both DE and HL should be even, to allow word transfers on the bus. BC must be even, transferring an even number of bytes, or the operation is undefined.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
V: Set if the result of decrementing BC is not equal to zero; cleared otherwise
N: Cleared
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	LDIW	11101101 11100000	3+r+w	L

LDIR
LOAD, INCREMENT AND REPEAT (BYTE)

LDIR

Operation:

```
repeat until (BC=0) begin
      (DE)        ←       (HL)
      DE          ←       DE + 1
      HL          ←       HL + 1
      BC(15-0)    ←       BC(15-0) – 1
      end
```

This instruction is used for block transfers of strings of data. The bytes of data at the location addressed by the HL register are loaded into memory starting at the location addressed by the DE register. The number of bytes moved is determined by the contents of the BC register. If the BC register contains zero when this instruction is executed, 65,536 bytes are transferred. The effect of incrementing the pointers during the transfer is important if the source and destination strings overlap with the source string starting at a higher memory address. Placing the pointers at the lowest address of the strings and incrementing the pointers ensures that the source string is copied without destroying the overlapping area.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value of the start of this instruction is saved before the interrupt request is accepted,so that the instruction can be properly resumed.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Cleared
V:	Cleared
N:	Cleared
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	LDIR	11101101 10110000	3+r+w	

LDIRW
LOAD, INCREMENT AND REPEAT (WORD)

LDIRW

Operation:

```
repeat until (BC=0) begin
    if (LW) then begin
        (DE)        ←    (HL)
        (DE+1)      ←    (HL+1)
        (DE+2)      ←    (HL+2)
        (DE+3)      ←    (HL+3)
        DE          ←    DE + 4
        HL          ←    HL + 4
        BC(15-0)    ←    BC(15-0) – 4
        end
    else begin
        (DE)        ←    (HL)
        (DE+1)      ←    (HL+1)
        DE          ←    DE + 2
        HL          ←    HL + 2
        BC(15-0)    ←    BC(15-0) – 2
        end
    end
```

This instruction is used for block transfers of strings of data. The words of data at the location addressed by the HL register are loaded into memory starting at the location addressed by the DE register. The number of words moved is determined by the contents of the BC register. If the BC register contains zero when this instruction is executed, 65,536 words are transferred. The effect of incrementing the pointers during the transfer is important if the source and destination strings overlap with the source string starting at a higher memory address. Placing the pointers at the lowest address of the strings and incrementing the pointers ensures that the source string is copied without destroying the overlapping area.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value of the start of this instruction is save before the interrupt request is accepted,so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
V: Cleared
N: Cleared
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	LDIRW	11101101 11110000	(3+r+w)n	L

MLT
MULTIPLY UNSIGNED (BYTE)

MLT R src = R

Operation: R(15-0) ← R(7-0) x R(15-8)

The contents of the upper byte of the source register are multiplied by the contents of the lower byte of the source register and the product is stored in the source register. Both operands. Both operands are treated as unsigned, binary integers.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Unaffected
V:	Unaffected
N:	Unaffected
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	MLT R	11101101 01rr1100	7	

Field Encodings: rr: 00 for BC, 01 for DE, 10 for HL, 11 for SP

MTEST
MODE TEST

MTEST

Operation:

S ← SR(7)
Z ← SR(6)
C ← SR(1)

The three mode control bits in the Select Register (SR) are transferred to the flags. This allows the program to determine the state of the machine.

Flags:

S: Set if Extended mode is in effect; cleared otherwise
Z: Set if Long word mode is in effect; cleared otherwise
H: Unaffected
V: Unaffected
N: Unaffected
C: Set if Lock mode is in effect; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	MTEST	11011101 11001111	2	

MULTW
MULTIPLY (WORD)

MULTW [HL,]src   src = R, RX, IM, X

Operation: HL(31-0) ← HL(15-0) x src(15-0)

The contents of the HL register are multiplied by the source operand and the product is stored in the HL register. The contents of the source are unaffected. Both operands are treated as signed, two's complement integers.

The initial contents of the HL register are overwritten by the result. The Carry flag is set to indicate that the upper word of the HL register is required to represent the result; if the Carry flag is cleared, the product can be correctly represented in 16 bits and the upper word of the HL register merely holds sign-extension data.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Unaffected
V: Cleared
N: Unaffected
C: Set if the product is less than –32768 or greater than or equal to 32768; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	MULTW [HL,]R	11101101 11001011 100100rr	10	
RX:	MULTW [HL,]RX	11101101 11001011 1001010y	10	
IM:	MULTW [HL,]nn	11101101 11001011 10010111 -n(low)- -n(high)	10	
X:	MULTW [HL,](XY+d)	11y11101 11001011 ——d— 10010010	12+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

MULTUW
MULTIPLY UNSIGNED (WORD)

MULTUW [HL,]src src = R, RX, IM, X

Operation: HL(31-0) ← HL(15-0) x src(15-0)

The contents of the HL register are multiplied by the source operand and the product is stored in the HL register. The contents of the source are unaffected. Both operands are treated as unsigned, binary integers.

The initial contents of the HL register are overwritten by the result. The Carry flag is set to indicate that the upper word of the HL register is required to represent the result; if the Carry flag is cleared, the product can be correctly represented in 16 bits and the upper word of the HL register merely holds zero.

Flags:

S: Cleared
Z: Set if the result is zero; cleared otherwise
H: Unaffected
V: Cleared
N: Unaffected
C: Set if the product is greater than or equal to 65536; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	MULTUW [HL,]R	11101101 11001011 100110rr	11	
RX:	MULTUW [HL,]RX	11101101 11001011 1001110y	11	
IM:	MULTUW [HL,]nn	11101101 11001011 10011111 -n(low)- -n(high)	11	
X:	MULTUW [HL,](XY+d)	11y11101 11001011 ——d— 10011010	13+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

NEG
NEGATE ACCUMULATOR

NEG [A]

Operation: A ← -A

The contents of the accumulator are negated, that is replaced by its two's complement value. Note that 80h is replaced by itself, because in two's complement representation the negative number with the greatest magnitude has no positive counterpart; for this case, the Overflow flag is set to 1.

Flags:

S:	Set if the result is negative; cleared otherwise
Z:	Set if the result is zero; cleared otherwise
H:	Set if there is a borrow from bit 4 of the result; cleared otherwise
V:	Set if the content of the accumulator was 80h before the operation; cleared otherwise
N:	Set
C:	Set if the content of the accumulator was not 00h before the operation; cleared if the content of the accumulator was 00h

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	NEG [A]	11101101 01000100	2	

NEGW
NEGATE HL REGISTER (WORD)

NEGW [HL]

Operation: HL(15-0) ← -HL(15-0)

The contents of the HL register are negated, that is replaced by its two's complement value. Note that 8000h is, replaced by itself, because in two's complement representation the negative number with the greatest magnitude has no positive counterpart; for this case, the Overflow flag is set to 1.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 4 of the result; cleared otherwise
V: Set if the content of the HL register was 8000h before the operation; cleared otherwise
N: Set
C: Set if the content of the HL register was not 0000h before the operation; cleared if the content of the HL register was 0000h

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	NEGW [HL]	11101101 01010100	2	

NOP
NO OPERATION

NOP

Operation: None

No operation.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	NOP	00000000	2	

OR
OR (BYTE)

OR [A,]src src = R, RX, IM, IR, X

Operation: A ← A OR src

A logical OR operation is performed between the corresponding bits of the source operand and the accumulator and the result is stored in the accumulator. A 1 bit is stored wherever either of the corresponding bits in the two operands is 1; otherwise a 0 bit is stored. The contents of the source are unaffected.

Flags:

S:	Set if the most significant bit of the result is set; cleared otherwise
Z:	Set if all bits of the result are zero; cleared otherwise
H:	Cleared
P:	Set if the parity is even; cleared otherwise
N:	Cleared
C:	Cleared

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	OR [A,]R	10110-r-	2	
RX:	OR [A,]RX	11y11101 1011010w	2	
IM:	OR [A,]n	11110110 ——n—	2	
IR:	OR [A,](HL)	10110110	2+r	
X:	OR [A,](XY+d)	11y11101 10110110 ——d—	4+r	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

ORW
OR (WORD)

ORW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) ← HL(15-0) OR src(15-0)

A logical OR operation is performed between the corresponding bits of the source operand and the HL register and the result is stored in the HL register. A 1 bit is stored wherever either of the corresponding bits in the two operands is 1; otherwise a 0 bit is stored. The contents of the source are unaffected.

Flags:

S:	Set if the most significant bit of the result is set; cleared otherwise
Z:	Set if all bits of the result are zero; cleared otherwise
H:	Cleared
P:	Set if the parity is even; cleared otherwise
N:	Cleared
C:	Cleared

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	ORW [HL,]R	11101101 101101rr	2	
RX:	ORW [HL,]RX	11y11101 10110111	2	
IM:	ORW [HL,]nn	11101101 10110110 -n(low) -n(high)-	2+r	
X:	ORW [HL,](XY+d)	11y11101 11110110 ——d—	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

OTDM
OUTPUT DECREMENT MEMORY

OTDM

Operation:

(C) ← (HL)
C ← C – 1
B ← B – 1
HL ← HL – 1

This instruction is used for block output of strings of data to on-chip peripherals. No external I/O transaction will be generated as a result of this instruction, although the I/O address will appear on the address bus and the write data will appear on the data bus while this internal write is occurring. The peripheral address is placed on the low byte of the address bus and zeros are placed on all other address lines. The byte of data from the memory location addressed by the HL register is loaded to the on-chip I/O port addressed by the C register. The C register, holding the port address, is decremented by one to select the next output port. The B register, used as a counter, is then decremented by one. The HL register is then decremented by one, thus moving the pointer to the next source for the output.

Flags:

S: Set if the result of decrementing B is negative; cleared otherwise
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Set if there is a borrow from bit 4 during the decrement of the B register; cleared otherwise
P: Set if the result of the decrement of the B register is even; cleared otherwise
N: Set if the most significant bit of the byte transferred was a 1; cleared otherwsie
C: Set if there is a borrow from the most significant bit during the decrement of the B register; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OTDM	11101101 10001011	2+r+o	

OTDMR
OUTPUT, DECREMENT MEMORY REPEAT

OTDMR

Operation:

```
repeat until (B=0) begin
    (C) ← (HL)
    C   ← C – 1
    B   ← B – 1
    HL  ← HL – 1
    end
```

This instruction is used for block output of strings of data to on-chip peripherals. No external I/O transaction will be generated as a result of this instruction, although the I/O address will appear on the address bus and the write data will appear on the data bus while this internal write is occurring. The peripheral address is placed on the low byte of the address bus and zeros are placed on all other address lines. The byte of data from the memory location addressed by the HL register is loaded to the on-chip I/O port addressed by the C register. The C register, holding the port address, is decremented by one to select the next output port. The B register, used as a counter, is then decremented by one. The HL register is then decremented by one, thus moving the pointer to the next source for the output. If the result of decrementing the B register is 0, the instruction is terminated, otherwise the output sequence is repeated. Note that if the B register contains 0 at the start of the execution of this instruction, 256 bytes are output.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S:	Cleared
Z:	Set
H:	Cleared
P:	Set
N:	Set if the most significant bit of the byte transferred was a 1; cleared otherwise
C:	Cleared

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OTDMR	11101101 10011011	2+r+o	

OTDR
OUTPUT, DECREMENT AND REPEAT (BYTE)

OTDR

Operation:

```
repeat until (B=0) begin
    B    ←  B – 1
    (C)  ←  (HL)
    HL   ←  HL – 1
    end
```

This instruction is used for block output of strings of data. The string of output data is loaded into the selected peripheral from memory at consecutive addresses, starting with the location addressed by the HL register and decreasing. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A(15-8) are not useable as part of a fixed port address. The decremented B register is used in the address.

First the B register, used as a counter, is decremented by one. The byte of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then decremented by one, thus moving the pointer to the next source for the output. If the result of decrementing the B register is 0, the instruction is terminated, otherwise the sequence is repeated. If the B register contains 0 at the start of the execution of this instruction, 256 bytes are output.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode

Syntax	Instruction Format	Execute Time	Note
OTDR	11101101 10111011	2+r+o	

OTDRW
OUTPUT, DECREMENT AND REPEAT (WORD)

OTDRW

Operation:

```
repeat until (BC=0) begin
    BC(15-0)   <-    BC(15-0) – 1
    (DE)       <-    (HL)
    HL         <-    HL – 2
    end
```

This instruction is used for block output of strings of data. The string of output data is loaded into the selected peripheral from memory at consecutive addresses, starting with the location addressed by the HL register and decreasing. During the I/O transaction the 32-bit DE register is placed on the address bus.

First the BC register, used as a counter, is decremented by one. The word of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then decremented by two, thus moving the pointer to the next source for the output. If the result of decrementing the BC register is 0, the instruction is terminated, otherwise the sequence is repeated. If the BC register contains 0 at the start of the execution of this instruction, 65536 bytes are output.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OTDRW	11101101 11111011	2+r+o	

OTIM
OUTPUT INCREMENT MEMORY

OTIM

Operation:

(C) ← (HL)
C ← C + 1
B ← B – 1
HL ← HL + 1

This instruction is used for block output of strings of data to on-chip peripherals. No external I/O transaction will be generated as a result of this instruction, although the I/O address will appear on the address bus and the write data will appear on the data bus while this internal write is occurring. The peripheral address is placed on the low byte of the address bus and zeros are placed on all other address lines. The byte of data from the memory location addressed by the HL register is loaded to the on-chip I/O port addressed by the C register. The C register, holding the port address, is incremented by one to select the next output port. The B register, used as a counter, is then decremented by one. The HL register is then incremented by one, thus moving the pointer to the next source for the output.

Flags:

S:	Set if the result of decrementing B is negative; cleared otherwise
Z:	Set if the result of decrementing B is zero; cleared otherwise
H:	Set if there is a borrow from bit 4 during the decrement of the B register; cleared otherwise
P:	Set if the result of the decrement of the B register is even; cleared otherwise
N:	Set if the most significant bit of the byte transferred was a 1; cleared otherwise
C:	Set if there is a borrow from the most significant bit during the decrement of the B register; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OTIM	11101101 10000011	2+r+o	

OTIMR
OUTPUT, INCREMENT MEMORY REPEAT

OTIMR

Operation:

```
repeat until (B=0) begin
    (C) ← (HL)
    C   ← C + 1
    B   ← B – 1
    HL  ← HL + 1
    end
```

This instruction is used for block output of strings of data to on-chip peripherals. No external I/O transaction will be generated as a result of this instruction, although the I/O address will appear on the address bus and the write data will appear on the data bus while this internal write is occurring. The peripheral address is placed on the low byte of the address bus and zeros are placed on all other address lines. The byte of data from the memory location addressed by the HL register is loaded to the on-chip I/O port addressed by the C register. The C register, holding the port address, is incremented by one to select the next output port. The B register, used as a counter, is then decremented by one. The HL register is then incremented by one, thus moving the pointer to the next source for the output. If the result of decrementing the B register is 0, the instruction is terminated, otherwise the output sequence is repeated. Note that if the B register contains 0 at the start of the execution of this instruction, 256 bytes are output.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Cleared
Z: Set
H: Cleared
P: Set
N: Set if the most significant bit of the byte transferred was a 1; cleared otherwsie
C: Cleared

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	OTIMR	11101101 10010011	2+r+o	

OTIR
OUTPUT, INCREMENT AND REPEAT (BYTE)

OTIR

Operation:

```
repeat until (B=0) begin
    B    ←  B – 1
    (C)  ←  (HL)
    HL   ←  HL + 1
    end
```

This instruction is used for block output of strings of data. The string of output data is loaded into the selected peripheral from memory at consecutive addresses, starting with the location addressed by the HL register and increasing. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A(15-8) are not useable as part of a fixed port address. The decremented B register is used in the address.

First the B register, used as a counter, is decremented by one. The byte of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then incremented by one, thus moving the pointer to the next source for the output. If the result of decrementing the B register is 0, the instruction is terminated, otherwise the sequence is repeated. If the B register contains 0 at the start of the execution of this instruction, 256 bytes are output.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OTIR	11101101 10110011	2+r+o	

OTIRW
OUTPUT, INCREMENT AND REPEAT (WORD)

OTIRW

Operation:

```
repeat until (BC=0) begin
    BC(15-0)    ←    BC(15-0) – 1
    (DE)        ←    (HL)
    HL          ←    HL + 2
    end
```

This instruction is used for block output of strings of data. The string of output data is loaded into the selected peripheral from memory at consecutive addresses, starting with the location addressed by the HL register and increasing. During the I/O transaction the 32-bit DE register is placed on the address bus.

First the BC register, used as a counter, is decremented by one. The word of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then incremented by two, thus moving the pointer to the next source for the output. If the result of decrementing the BC register is 0, the instruction is terminated, otherwise the sequence is repeated. If the BC register contains 0 at the start of the execution of this instruction, 65536 bytes are output.

This instruction can be interrupted after each execution of the basic operation. The Program Counter value at the start of this instruction is saved before the interrupt request is accepted, so that the instruction can be properly resumed.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OTIRW	11101101 11110011	2+r+o	

OUT
OUTPUT (BYTE)

OUT (C),src   src = R, IM

Operation: (C) ← src

The byte of data from the source is loaded into the selected peripheral. During the I/O transaction, the contents of the 32-bit BC register are placed on the address bus.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	OUT (C),R	11101101 01 -r- 001	3+o	
IM:	OUT (C),n	11101101 01110001 —n—	3+o	

Field Encodings: r: per convention

OUTW
OUTPUT (WORD)

OUTW (C),src   src = R, IM

Operation: (C) ← src(15-0)

The word of data from the source is loaded into the selected peripheral. During the I/O transaction, the contents of the 32-bit BC register are placed on the address bus.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	OUTW (C),R	11011101 01rrr 001	2+o	
IM:	OUTW (C),nn	11111101 01111001 -n(low)- -n(high)	2+o	

Field Encodings: rrr: 000 for BC, 010 for DE, 111 for HL

OUT
OUTPUT ACCUMULATOR

OUT (n),A

Operation: (n) ← A

The byte of data from the accumulator is loaded into the selected peripheral. During the I/O transaction, the 8-bit peripheral address from the instruction is placed on the low byte of the address bus, the contents of the accumulator are placed on address lines A(15-8), and the high-order address lines are all zeros.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OUT (n),A	11010011 ——n—	3+o	

OUT0
OUTPUT (TO PAGE 0)

OUT0 (n),src   src = R

Operation: (n) ← src

The byte of data from the source register is loaded into the selected on-chip peripheral. No external I/O transaction will be generated as a result of this instruction, although the I/O address will appear on the address bus and the write data will appear on the data bus while this internal write is occurring. The peripheral address is placed on the low byte of the address bus and zeros are placed on all other address lines.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	OUT0 (n),R	11101101 00-r-001 ——n—		3+o

Field Encodings: r: per convention

OUTA
OUTPUT DIRECT TO PORT ADDRESS (BYTE)

OUT (nn),A

Operation: (nn) ← A

The byte of data from the accumulator is loaded into the selected peripheral. During the I/O transaction, the peripheral address from the instruction is placed on the address bus. Any bytes of address not specified in the instruction are driven on the address lines are all zeros.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OUTA (nn),A	11101101 11010011 -n(low)- -n(high)	2+o	I

OUTAW
OUTPUT DIRECT TO PORT ADDRESS (WORD)

OUT (nn),HL

Operation: (nn)← HL(15-0)

The word of data from the HL register is loaded into the selected peripheral. During the I/O transaction, the peripheral address from the instruction is placed on the address bus. Any bytes of address not specified in the instruction are driven on the address lines are all zeros.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Unaffected
V:	Unaffected
N:	Unaffected
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OUTAW (nn),HL	11111101 11010011 -n(low)- -n(high)	2+o	I

OUTD
OUTPUT AND DECREMENT (BYTE)

OUTD

Operation:

B ← B - 1
(C) ← (HL)
HL ← HL - 1

This instruction is used for block output of strings of data. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A15-A8 are not useable as part of a fixed port address. The decremented B register is used in the address.

First the B register, used as a counter, is decremented by one. The byte of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then decremented by one, thus moving the pointer to the next source for the output.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode

Syntax	Instruction Format	Execute Time	Note
OUTD	11101101 10101011	2+r+o	

OUTDW
OUTPUT AND DECREMENT (WORD)

OUTDW

Operation:

BC(15-0) ← BC(15-0) - 1
(DE) ← (HL)
HL ← HL - 2

This instruction is used for block output of strings of data. During the I/O transaction the 32-bit DE register is placed on the address bus.

First the BC register, used as a counter, is decremented by one. The word of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then decremented by two, thus moving the pointer to the next source for the output.

Flags:

S:	Unaffected
Z:	Set if the result of decrementing BC is zero; cleared otherwise
H:	Unaffected
V:	Unaffected
N:	Set
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OUTDW	11101101 11101011	2+r+o	

OUTI
OUTPUT AND INCREMENT (BYTE)

OUTI

Operation:

B ← B - 1
(C) ← (HL)
HL ← HL + 1

This instruction is used for block output of strings of data. During the I/O transaction the 32-bit BC register is placed on the address bus. Note that the B register contains the loop count for this instruction so that A15-A8 are not useable as part of a fixed port address. The decremented B register is used in the address.

First the B register, used as a counter, is decremented by one. The byte of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then incremented by one, thus moving the pointer to the next source for the output.

Flags:

S: Unaffected
Z: Set if the result of decrementing B is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	OUTI	11101101 10100011	2+r+o	

OUTIW
OUTPUT AND INCREMENT (WORD)

OUTIW

Operation:

BC(15-0) ← BC(15-0) –1
(DE) ← (HL)
HL ← HL + 2

This instruction is used for block output of strings of data. During the I/O transaction the 32-bit DE register is placed on the address bus.

First the BC register, used as a counter, is decremented by one. The word of data from the memory location addressed by the HL register is loaded into the selected peripheral. The HL register is then incremented by two, thus moving the pointer to the next source for the output.

Flags:

S: Unaffected
Z: Set if the result of decrementing BC is zero; cleared otherwise
H: Unaffected
V: Unaffected
N: Set
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	OUTIW	11101101 11100011	2+r+o	

POP
POP ACCUMULATOR

POP dst dst = AF

Operation:

```
F   ← (SP)
A   ← (SP+1)
SP  ← SP + 2
if (LW) then begin
    SP ← SP + 2
    end
```

The contents of the memory location addressed by the Stack Pointer (SP) are loaded into the destination in ascending byte order from ascending address memory locations. For this instruction, the Flag register is the least significant byte, followed by the Accumulator. The SP is then incremented by two (by four in the Long Word mode). Note that in the Long Word mode only one word is read from memory, although the SP is in fact incremented by four.

Flags:

S: Loaded from (SP)
Z: Loaded from (SP)
H: Loaded from (SP)
V: Loaded from (SP)
N: Loaded from (SP)
C: Loaded from (SP)

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	POP AF	11110001	2+r	L

POP
POP CONTROL REGISTER

POP dst dst = SR

Operation:

```
if (LW) then begin
    dst(6-0)   ←  (SP)
    dst(15-8)  ←  (SP+1)
    dst(23-16) ←  (SP+2)
    dst(31-24) ←  (SP+3)
    SP         ←  SP + 4
    end
else begin
    dst(6-0)   ←  (SP)
    dst(15-8)  ←  (SP+1)
    dst(23-16) ←  (SP+1)
    dst(31-24) ←  (SP+1)
    SP         ←  SP + 2
    end
```

The contents of the memory location addressed by the Stack Pointer (SP) are loaded into the destination in ascending byte order from ascending address memory locations. The SP is then incremented by two (by four in the Long Word mode). Note that when not in the Long Word mode the most significant byte read from memory is also written to the two most significant bytes of the SR. Also note that the XM bit is unaffected by this instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	POP SR	11101101 11000001	3+r	L

POP
POP REGISTER

POP dst dst = R, RX

Operation:

```
if (LW) then begin
    dst(7-0 )   ←   (SP)
    dst(15-8)   ←   (SP+1)
    dst(23-16)  ←   (SP+2)
    dst(31-24)  ←   (SP+3)
    SP          ←   SP + 4
    end
else begin
    dst(7-0)    ←   (SP)
    dst(15-8)   ←   (SP+1)
    SP          ←   SP + 2
    end
```

The contents of the memory location addressed by the Stack Pointer (SP) are loaded into the destination in ascending byte order from ascending address memory locations. The SP is then incremented by two (by four in the Long Word mode).

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	POP R	11rr 0001	1+r	L
RX:	POP RX	11y11101 11100001	1+r	L

Field Encodings: rr: 00 for BC, 01 for DE, 10 for HL
y: 0 for IX, 1 for IY

PUSH
PUSH ACCUMULATOR

PUSH src   src = AF

Operation:

```
if (LW) then begin
    SP      ←  SP - 4
    (SP)    ←  F
    (SP+1)  ←  A
    (SP+2)  ←  00h
    (SP+3)  ←  00h
    end
else begin
    SP      ←  SP - 2
    (SP)    ←  F
    (SP+1)  ←  A
    end
```

The Stack Pointer (SP) is decremented by two (by four in Long Word mode) and the source is loaded into the memory locations addressed by the SP in ascending byte order in ascending address memory locations. For this instruction, the Flag register is the least significant byte, followed by the Accumulator. The other two bytes written in the Long Word mode are all zeros. The Flag register and Accumulator are unaffected.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	PUSH AF	11110101	3+w	L

PUSH
PUSH CONTROL REGISTER

PUSH src src = SR

Operation:

```
if (LW) then begin
    SP      ←  SP - 4
    (SP)    ←  src(7-0)
    (SP+1)  ←  src(15-8)
    (SP+2)  ←  src(23-16)
    (SP+3)  ←  src(31-24)
    end
else begin
    SP      ←  SP - 2
    (SP)    ←  src(7-0)
    (SP+1)  ←  src(15-8)
    end
```

The Stack Pointer (SP) is decremented by two (by four in Long Word mode) and the source is loaded into the memory locations addressed by the SP in ascending byte order in ascending address memory locations. The contents of the source are unaffected.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	PUSH SR	11101101 11000101	3+w	L

PUSH
PUSH IMMEDIATE

PUSH src   src = IM

Operation:

```
if (LW) then begin
    SP      ←  SP - 4
    (SP)    ←  src(7-0)
    (SP+1)  ←  src(15-8)
    (SP+2)  ←  src(23-16)
    (SP+3)  ←  src(31-24)
    end
else begin
    SP      ←  SP - 2
    (SP)    ←  src(7-0)
    (SP+1)  ←  src(15-8)
    end
```

The Stack Pointer (SP) is decremented by two (by four in Long Word mode) and the source is loaded into the memory locations addressed by the SP in ascending byte order in ascending address memory locations.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
IM:	PUSH nn	11111101 11110101 -n(low)- -n(high)	3+w	I, L

PUSH
PUSH REGISTER

PUSH src   src = R, RX

Operation:

```
if (LW) then begin
    SP      ←  SP - 4
    (SP)    ←  src(7-0)
    (SP+1)  ←  src(15-8)
    (SP+2)  ←  src(23-16)
    (SP+3)  ←  src(31-24)
    end
else begin
    SP      ←  SP - 2
    (SP)    ←  src(7-0)
    (SP+1)  ←  src(15-8)
    end
```

The Stack Pointer (SP) is decremented by two (by four in Long Word mode) and the source is loaded into the memory locations addressed by the SP in ascending byte order in ascending address memory locations. The contents of the source are unaffected.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	PUSH R	11rr0101	3+w	L
RX:	PUSH RX	11y11101 11100101	3+w	L

Field Encodings:
rr: 00 for BC, 01 for DE, 10 for HL
y: 0 for IX, 1 for IY

RES
RESET BIT

RES b, dst   dst = R, IR, X

Operation: dst(b) ← 0

The specified bit b within the destination operand is cleared to 0. The other bits in the destination are unaffected. The bit to be reset is specified by a 3-bit field in the instruction; this field contains the binary encoding for the bit number to be cleared. The bit number b must be between 0 and 7.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	RES b,R	11001011 10bbb -r-	2	
IR:	RES b,(HL)	11001011 10bbb110	2+r	
X:	RES b,(XY+d)	11y11101 11001011 ——d— 10bbb110	4+r	I

Field Encodings: r: per convention
y: 0 for IX, 1 for IY

RESC
RESET CONTROL BIT

RESC mode   mode = LCK, LW

Operation:

```
if (mode = LCK) then begin
    SR(1)  ← 0
    end
else begin
    SR(6)  ← 0
    end
```

When reseting Lock mode (LCK), the LCK bit (bit 1) in the Select Register (SR) is set to 0, enabling external bus requests. Note that these requests cannot be granted until after the instruction has been executed, and that one or more of the succeeding instructions may also have been fetched for decoding before this instruction has been executed.

When reseting Long Word mode (LW), the LW bit (bit 6) in the SR is set to 0, selecting 16-bit words. When using 16-bit words, all word load operations transfer 16 bits.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Unaffected
V:	Unaffected
N:	Unaffected
C:	Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	RESC mode	11mm1101 11111111	4	

Field Encodings: mm: 01 for LW, 10 for LCK

RET
RETURN

RET [cc]

Operation:

```
if (cc is TRUE) then begin
    if (XM) then begin
        PC(7-0)     ←    (SP)
        PC(15-8)    ←    (SP+1)
        PC(23-16)   ←    (SP+2)
        PC(31-24)   ←    (SP+3)
        SP          ←    SP + 4
        end
    else begin
        PC(7-0)     ←    (SP)
        PC(15-8)    ←    (SP+1)
        SP          ←    SP + 2
        end
    end
```

This instruction is used to return to a previously executing procedure at the end of a procedure entered by a Call instruction. For a conditional return, one of the Zero, Carry, Sign, or Parity/Overflow flags is checked to see if its setting matches the condition code "cc" encoded in the instruction; if the condition is not satisfied, the instruction following the Return instruction is executed, otherwise a value is popped from the stack and loaded into the Program Counter (PC), thereby specifying the location of the next instruction to be executed. For an unconditional return, the return is always taken and a condition code is not specified.

This instruction is also used to return to a previously executing procedure at the end of a procedure entered by an interrupt in the assigned vectors mode, if Z80 family peripherals are used external to the Z380 MPU.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	RET CC	11-cc000	note	X
	RET	11001001	2+r	X

Field Encodings: cc: 000 for NZ, 001 for Z, 010 for NC, 011 for C,
100 for PO/NV, 101 for PE/V, 110 for P/NS, 111 for M/S

Note: 2 if CC is false, 2+r if CC is true

RETB
RETURN FROM BREAKPOINT

Operation: PC (31-0) ← SPC (31-0)

This instruction is used to return to a previously executing procedure at the end of a breakpoint. The contents of the Shadow Program Counter (SPC), which holds the address of the next instruction of the previously executing procedure, are loaded into the Program Counter (PC).

Note that maskable interrupts (if IEF1 is set) and non-maskable interrupt are enabled after the instruction following RETB is executed.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode

Syntax	Instruction Format	Execute Time	Note
RETB	11101101 01010101	2	

RETI
RETURN FROM INTERRUPT

RETI

Operation:

```
if (XM) then begin
    PC(7-0)    ←   (SP)
    PC(15-8)   ←   (SP+1)
    PC(23-16)  ←   (SP+2)
    PC(31-24)  ←   (SP+3)
    SP         ←   SP + 4
    end
else begin
    PC(7-0)    ←   (SP)
    PC(15-8)   ←   (SP+1)
    SP         ←   SP + 2
    end
```

This instruction is used to return to a previously executing procedure at the end of a procedure entered by an interrupt. The contents of the location addressed by the Stack Pointer (SP) are popped into the Program Counter (PC), thereby specifying the location of the next instruction to be executed. A special sequence of bus transactions is performed when this instruction is executed in order to control Z80 family peripherals; see the description of the external interface for more details.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	RETI	11101101 01001101	2+r	X

RETN
RETURN FROM NONMASKABLE INTERRUPT

RETN

Operation:

```
if (XM) then begin
    PC(7-0)    ←    (SP)
    PC(15-8)   ←    (SP+1)
    PC(23-16)  ←    (SP+2)
    PC(31-24)  ←    (SP+3)
    SP         ←    SP + 4
    end
else begin
    PC(7-0)    ←    (SP)
    PC(15-8)   ←    (SP+1)
    SP         ←    SP + 2
    end
IEF1           ←    IEF2
```

This instruction is used to return to a previously executing procedure at the end of a procedure entered by a nonmaskable interrupt. The contents of the location addressed by the Stack Pointer (SP) are popped into the Program Counter (PC), thereby specifying the location of the next instruction to be executed. The previous setting of the interrupt enable bit is restored by execution of this instruction.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode

Syntax	Instruction Format	Execute Time	Note
RETN	11101101 01000101	2+r	X

5

RL
ROTATE LEFT (BYTE)

RL dst dst = R, IR, X

Operation:

tmp ← dst
dst(0) ← C
C ← dst(7)
dst(n+1) ← tmp(n) for n = 0 to 6

The contents of the destination operand are concatenated with the Carry flag and together they are rotated left one bit position. Bit 7 of the destination operand is moved to the Carry flag and the Carry flag is moved to bit 0 of the destination.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from bit 7 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	RL R	11001011 00010-r-	2	
IR:	RL (HL)	11001011 00010110	2+r	
X:	RL (XY+d)	11y11101 11001011 ——d— 00010110	4+r	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY

RLW
ROTATE LEFT (WORD)

RLW dst dst = R, RX, IR, X

Operation:

```
tmp         ←  dst
dst(0)      ←  C
C           ←  dst(15)
dst(n+1)    ←  tmp(n) for n = 0 to 14
```

The contents of the destination operand are concatenated with the Carry flag and together they are rotated left one bit position. The most significant bit of the destination operand is moved to the Carry flag and the Carry flag is moved to bit 0 of the destination.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from the most significant bit was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	RLW R	11101101 11001011 000100rr	2	
RX:	RLW RX	11101101 11001011 0001010y	2	
IR:	RLW (HL)	11101101 11001011 00010010	2+r	
X:	RLW (XY+d)	11y11101 11001011 ——d— 00010010	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

RLA
ROTATE LEFT (ACCUMULATOR)

RLA

Operation:

tmp ← A
A(0) ← C
C ← A(7)
A(n+1) ← tmp(n) for n = 0 to 6

The contents of the accumulator are concatenated with the Carry flag and together they are rotated left one bit position. Bit 7 of the accumulator is moved to the Carry flag and the Carry flag is moved to bit 0 of the accumulator.

Flags:

S:	Unaffected
Z:	Unaffected
H:	Cleared
P:	Unaffected
N:	Cleared
C:	Set if the bit rotated from bit 7 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	RLA	00010111	2	

RLC
ROTATE LEFT CIRCULAR (BYTE)

RLC dst dst = R, IR, X

Operation:

```
tmp        ←  dst
C          ←  dst(7)
dst(0)     ←  tmp(7)
dst(n+1)   ←  tmp(n) for n = 0 to 6
```

The contents of the destination operand are rotated left one bit position. Bit 7 of the destination operand is moved to the bit 0 position and also replaces the Carry flag.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from bit 7 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	RLC R	11001011 00000-r-	2	
IR:	RLC (HL)	11001011 00000110	2+r	
X:	RLC (XY+d)	11y11101 11001011 ——d— 00000110	4+r	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY

RLCW
ROTATE LEFT CIRCULAR (WORD)

RLCW dst dst = R, RX, IR, X

Operation:

tmp ← dst
C ← dst(15)
dst(0) ← tmp(15)
dst(n+1) ← tmp(n) for n = 0 to 14

The contents of the destination operand are rotated left one bit position. The most significant bit of the destination operand is moved to the bit 0 position and also replaces the Carry flag.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from the most significant bit was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	RLCW R	11101101 11001011 000000rr	2	
RX:	RLCW RX	11101101 11001011 0000010y	2	
IR:	RLCW (HL)	11101101 11001011 00000010	2+r	
X:	RLCW (XY+d)	11y11101 11001011 ——d— 00000010	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

RLCA
ROTATE LEFT CIRCULAR (ACCUMULATOR)

RLCA

Operation:

tmp ← A
C ← A(7)
A(0) ← tmp(7)
A(n+1) ← tmp(n) for n = 0 to 6

The contents of the accumulator are rotated left one bit position. Bit 7 of the accumulator is moved to the bit 0 position and also replaces the Carry flag.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
P: Unaffected
N: Cleared
C: Set if the bit rotated from bit 7 was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	RLCA	00000111	2	

RLD
ROTATE LEFT DIGIT

RLD

Operation:

```
tmp(3-0)  ←  A(3-0)
A(3-0)    ←  dst(7-4)
dst(7-4)  ←  dst(3-0)
dst(3-0)  ←  tmp(3-0)
```

The low digit of the accumulator is logically concatenated to the destination byte whose memory address is in the HL register. The resulting three-digit quantity is rotated to the left by one BCD digit (four bits). The lower digit of the source is moved to the upper digit of the source; the upper digit of the source is moved to the lower digit of the accumulator, and the lower digit of the accumulator is moved to the lower digit of the source. The upper digit of the accumulator is unaffected. In multiple-digit BCD arithmetic, this instruction can be used to shift to the left a string of BCD digits, thus multiplying it by a power of ten. The accumulator serves to transfer digits between successive bytes of the string. This is analogous to the use of the Carry flag in multiple-precision shifting using the RL instruction.

Flags:

S:	Set if the accumulator is negative after the operation; cleared otherwise
Z:	Set if the accumulator is zero after the operation; cleared otherwise
H:	Cleared
P:	Set if the parity of the accumulator is even after the operation; cleared otherwise
N:	Cleared
C:	Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	RLD	11101101 01101111	3+r	

RR
ROTATE RIGHT (BYTE)

RR dst dst = R, IR, X

Operation:

```
tmp     ←  dst
dst(7)  ←  C
C       ←  dst(0)
dst(n)  ←  tmp(n+1) for n = 0 to 6
```

The contents of the destination operand are concatenated with the Carry flag and together they are rotated right one bit position. Bit 0 of the destination operand is moved to the Carry flag and the Carry flag is moved to bit 7 of the destination.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from bit 0 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	RR R	11001011 00011-r-	2	
IR:	RR (HL)	11001011 00011110	2+r	
X:	RR (XY+d)	11y11101 11001011 ——d— 00011110	4+r	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY

RRW
ROTATE RIGHT (WORD)

RRW dst   dst = R, RX, IR, X

Operation:

tmp ← dst
C ← dst(0)
dst(15) ← C
dst(n) ← tmp(n+1) for n = 0 to 14

The contents of the destination operand are concatenated with the Carry flag and together they are rotated right one bit position. Bit 0 of the destination operand is moved to the Carry flag and the Carry flag is moved to the most significant bit of the destination.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from bit 0 was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	RRW R	11101101 11001011 000110rr	2	
RX:	RRW RX	11101101 11001011 0001110y	2	
IR:	RRW (HL)	11101101 11001011 00011010	2+r	
X:	RRW (XY+d)	11y11101 11001011 ——d— 00011010	4+r	I

Field Encodings:
rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

RRA
ROTATE RIGHT (ACCUMULATOR)

RRA

Operation:

tmp ← A
A(7) ← C
C ← A(0)
A(n) ← tmp(n+1) for n = 0 to 6

The contents of the accumulator are concatenated with the Carry flag and together they are rotated right one bit position. Bit 0 of the accumulator is moved to the Carry flag and the Carry flag is moved to bit 7 of the accumulator.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
P: Unaffected
N: Cleared
C: Set if the bit rotated from bit 0 was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	RRA	00011111	2	

RRC
ROTATE RIGHT CIRCULAR (BYTE)

RRC dst dst = R, IR, X

Operation:

```
tmp     ← dst
C       ← dst(0)
dst(7)  ← tmp(0)
dst(n)  ← tmp(n+1) for n = 0 to 6
```

The contents of the destination operand are rotated right one bit position. Bit 0 of the destination operand is moved to the bit 7 position and also replaces the Carry flag.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from bit 0 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	RRC R	11001011 00001-r-	2	
IR:	RRC (HL)	11001011 00001110	2+r	
X:	RRC (XY+d)	11y11101 11001011 ——d— 00001110	4+r	I

Field Encodings: r: per convention
y: 0 for IX, 1 for IY

RRCW
ROTATE RIGHT CIRCULAR (WORD)

RRCW dst   dst = R, RX, IR, X

Operation:

tmp ← dst
C ← dst(0)
dst(15) ← tmp(0)
dst(n) ← tmp(n+1) for n = 0 to 14

The contents of the destination operand are rotated right one bit position. Bit 0 of the destination operand is moved to the most significant bit position and also replaces the Carry flag.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit rotated from bit 0 was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	RRCW R	11101101 11001011 000010rr	2	
RX:	RRCW RX	11101101 11001011 0000110y	2	
IR:	RRCW (HL)	11101101 11001011 00001010	2+r	
X:	RRCW (XY+d)	11y11101 11001011 ——d— 00001010	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

RRCA
ROTATE RIGHT CIRCULAR (ACCUMULATOR)

RRCA

Operation:

tmp ← A
C ← A(0)
A(7) ← tmp(0)
A(n) ← tmp(n+1) for n = 0 to 6

The contents of the accumulator are rotated right one bit position. Bit 0 of the accumulator is moved to the bit 7 position and also replaces the Carry flag.

Flags:

S: Unaffected
Z: Unaffected
H: Cleared
P: Unaffected
N: Cleared
C: Set if the bit rotated from bit 0 was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	RRCA	00001111	2	

RRD
ROTATE RIGHT DIGIT

RRD

Operation:

tmp(3-0) ← A(3-0)
A(3-0) ← dst(3-0)
dst(3-0) ← dst(7-4)
dst(7-4) ← tmp(3-0)

The low digit of the accumulator is logically concatenated to the destination byte whose memory address is in the HL register. The resulting three-digit quantity is rotated to the right by one BCD digit (four bits). The upper digit of the source is moved to the lower digit of the source; the lower digit of the source is moved to the lower digit of the accumulator, and the lower digit of the accumulator is moved to the upper digit of the source. The upper digit of the accumulator is unaffected. In multiple-digit BCD arithmetic, this instruction can be used to shift to the right a string of BCD digits, thus dividing it by a power of ten. The accumulator serves to transfer digits between successive bytes of the string. This is analogous to the use of the Carry flag in multiple-precision shifting using the RR instruction.

Flags:

S: Set if the accumulator is negative after the operation; cleared otherwise
Z: Set if the accumulator is zero after the operation; cleared otherwise
H: Cleared
P: Set if the parity of the accumulator is even after the operation; cleared otherwise
N: Cleared
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	RRD	11101101 01100111	3+r	

RST
RESTART

RST address

Operation:

```
if (XM) then begin
    SP      ←  SP - 4
    (SP)    ←  PC(7-0)
    (SP+1)  ←  PC(15-8)
    (SP+2)  ←  PC(23-16)
    (SP+3)  ←  PC(31-24)
    end
else begin
    SP      ←  SP - 2
    (SP)    ←  PC(7-0)
    (SP+1)  ←  PC(15-8)
    end
PC          ← address
```

The current Program Counter (PC) is pushed onto the stack and the PC is loaded with a constant address encoded in the instruction. Execution then begins at this address. The restart instruction allows for a call to one of eight fixed locations as shown in the table below. The table also indicates the encoding of the address used in the instruction encoding. (The address is in hexadecimal, the encoding in binary.)

Address	**t encoding**
00000000h	000
00000008h	001
00000010h	010
00000018h	011
00000020h	100
00000028h	101
00000030h	110
00000038h	111

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	RST address	11-t-111	4+w	X

Field Encodings: 000 for 00h, 001 for 08h, 010 for 10h, 011 for 18h, 100 for 20h, 101 for 28h, 110 for 30h, 111 for 38h

SBC
SUBTRACT WITH CARRY (BYTE)

SBC A,src src = R, RX, IM, IR, X

Operation: A ← A - src - C

The source operand together with the Carry flag is subtracted from the accumulator and the difference is stored in the accumulator. The contents of the source are unaffected. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 4 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the operands are of different signs and the result is of the same sign as the source; cleared otherwise
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SBC A,R	10011-r-	2	
RX:	SBC A,RX	11y11101 1001110w	2	
IM:	SBC A,n	11011110 ——n—	2	
IR:	SBC A,(HL)	10011110	2+r	
X:	SBC A,(XY+d)	11y11101 10011110 ——d—	4+r	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

SBC
SUBTRACT WITH CARRY (WORD)

SBC HL,src dst = HL
src = BC, DE, HL, SP

Operation: HL(15-0) ← HL(15-0) - src(15-0) - C

The source operand together with the Carry flag is subtracted from the HL register and the difference is stored in the HL register. The contents of the source are unaffected. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 12 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the operands are of different signs and the result is of the same sign as the the source; cleared otherwise
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SBC HL,R	11101101 01rr0010	2	

Field Encodings: rr: 00 for BC, 01 for DE, 10 for HL, 11 for SP

SBCW
SUBTRACT WITH CARRY (WORD)

SBCW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) ← HL(15-0) - src(15-0) - C

The source operand together with the Carry flag is subtracted from the HL register and the difference is stored in the HL register. The contents of the source are unaffected. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 12 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the operands are of different signs and the result is of the same sign as the source; cleared otherwise
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SBCW [HL,]R	11101101 100111rr	2	
RX:	SBCW [HL,]RX	11y11101 10011111	2	
IM:	SBCW [HL,]nn	11101101 10011110 -n(low) -n(high)-	2	
X:	SBCW [HL,](XY+d)	11y11101 11011110 ——d—	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

SCF
SET CARRY FLAG

SCF

Operation: C ← 1

The Carry flag is set to 1.

Flags:
S: Unaffected
Z: Unaffected
H: Cleared
V: Unaffected
N: Cleared
C: Set

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	SCF	00110111	2	

SET
SET BIT

SET b, dst   dst = R, IR, X

Operation: dst(b) ← 1

The specified bit b within the destination operand is set to 1. The other bits in the destination are unaffected. The bit to be set is specified by a 3-bit field in the instruction; this field contains the binary encoding for the bit number to be set. The bit number b must be between 0 and 7.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	SET b,R	11001011 11bbb -r-	2	
IR:	SET b,(HL)	11001011 11bbb110	2+r	
X:	SET b,(XY+d)	11y11101 11001011 ——d— 11bbb110	4+r	I

Field Encodings:

r:	per convention
y:	0 for IX, 1 for IY

SETC
SET CONTROL BIT

SETC mode   mode = LCK, LW, XM

Operation:

```
if (mode = LCK) then begin
    SR(1)  ←  1
    end
else if (mode = LW) then begin
    SR(6)  ←  1
    end
else begin
    SR(7)  ←  1
    end
```

When setting Lock mode (LCK), the LCK bit (bit 1) in the Select Register (SR) is set to 1, disabling external bus requests. Note that bus requests are not disabled until after this instruction has been executed, and that one or more of the succeeding instructions may also have been fetched for decoding before this instruction has been executed.

When setting Long Word mode (LW), the LW bit (bit 6) in the SR is set to 1, selecting 32-bit words. When using 32-bit words, all word load instructions transfer 32 bits.

When setting Extended mode (XM), the XM bit (bit 7) in the SR is set to 1, selecting addresses modulo 4,294,967,296 (32 bits) as opposed to addresses modulo 65536 (16 bits) in Native mode. In Extended mode CALL and RETurn instructions save and restore 32 bit PC values to and from the stack, and the PC pushed to the stack in response to an interrupt is 32 bits. In Extended mode, address manipulation instructions such as INCrement, DECrement, ADD, and Jump Relative (JR) employ 32-bit addresses. Note that it is not possible to exit from Extended mode except via reset.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	SETC mode	11mm1101 11110111	4	

Field Encodings: mm: 01 for LW, 10 for LCK, 11 for XM

SLA
SHIFT LEFT ARITHMETIC (BYTE)

SLA dst dst = R, IR, X

Operation:

tmp ← dst
C ← dst(7)
dst(0) ← 0
dst(n+1) ← tmp(n) for n = 0 to 6

The contents of the destination operand are shifted left one bit position. Bit 7 of the destination operand is moved to the Carry flag and zero is shifted into bit 0 of the destination.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit shifted from bit 7 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	SLA R	11001011 00100-r-	2	
IR:	SLA (HL)	11001011 00100110	2+r	
X:	SLA (XY+d)	11y11101 11001011 ——d— 00100110	4+r	I

Field Encodings: r: per convention
y: 0 for IX, 1 for IY

SLAW
SHIFT LEFT ARITHMETIC (WORD)

SLAW dst dst = R, RX, IR, X

Operation:

tmp ← dst
dst(0) ← 0
C ← dst(15)
dst(n+1) ← tmp(n) for n = 0 to 14

The contents of the destination operand are shifted left one bit position. The most significant bit of the destination operand is moved to the Carry flag and zero is shifted into bit 0 of the destination.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit shifted from the most significant bit was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SLAW R	11101101 11001011 001000rr	2	
RX:	SLAW RX	11101101 11001011 0010010y	2	
IR:	SLAW (HL)	11101101 11001011 00100010	2+r	
X:	SLAW (XY+d)	11y11101 11001011 ——d— 00100010	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

SLP
SLEEP

SLP

Operation:

```
if (STBY not enabled) then
    CPU Halts
else
    Z380 enters Standby mode
```

With Standby mode disabled, this instruction is interpreted and executed as a HALT instruction.

With Standby mode enabled, executing this instruction causes all device operation to stop, thus minimizing power dissipation. The /STNBY signal is asserted to indicate this Standby mode status. /STNBY remains asserted until an interrupt or reset request is accepted, which causes the device to exit Standby mode. If the option is enabled, an external bus request also causes the devcie to exit the Standby mode.

Flags:

S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
	SLP	11101101 01110110	2	

SRA
SHIFT RIGHT ARITHMETIC (BYTE)

SRA dst   dst = R, IR, X

Operation:

```
tmp     ←  dst
C       ←  dst(0)
dst(7)  ←  tmp(7)
dst(n)  ←  tmp(n+1) for n = 0 to 6
```

The contents of the destination operand are shifted right one bit position. Bit 0 of the destination operand is moved to the Carry flag and bit 7 remains unchanged.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit shifted from bit 0 was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SRA R	11001011 00101-r-	2	
IR:	SRA (HL)	11001011 00101110	2+r	
X:	SRA (XY+d)	11y11101 11001011 ——d— 00101110	4+r	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY

SRAW
SHIFT RIGHT ARITHMETIC (WORD)

SRAW dst dst = R, RX, IR, X

Operation:

tmp ← dst
C ← dst(0)
dst(15) ← tmp(15)
dst(n) ← tmp(n+1) for n = 0 to 14

The contents of the destination operand are shifted right one bit position. Bit 0 of the destination operand is moved to the Carry flag and the most significant bit remains unchanged.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit shifted from bit 0 was a 1; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SRAW R	11101101 11001011 001010rr	2	
RX:	SRAW RX	11101101 11001011 0010110y	2	
IR:	SRAW (HL)	11101101 11001011 00101010	2+r	
X:	SRAW (XY+d)	11y11101 11001011 ——d— 00101010	4+r	I

Field Encodings:
rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

SRL
SHIFT RIGHT LOGICAL (BYTE)

SRL dst dst = R, IR, X

Operation:

tmp ← dst
C ← dst(0)
dst(7) ← 0
dst(n) ← tmp(n+1) for n = 0 to 6

The contents of the destination operand are shifted right one bit position. Bit 0 of the destination operand is moved to the Carry flag and zero is shifted into bit 7 of the destination.

Flags:

S:	Cleared
Z:	Set if the result is zero; cleared otherwise
H:	Cleared
P:	Set if parity of the result is even; cleared otherwise
N:	Cleared
C:	Set if the bit shifted from bit 0 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	SRL R	11001011 00111-r-	2	
IR:	SRL (HL)	11001011 00111110	2+r	
X:	SRL (XY+d)	11y11101 11001011 ——d— 00111110	4+r	I

Field Encodings: r: per convention
y: 0 for IX, 1 for IY

SRLW
SHIFT RIGHT LOGICAL (WORD)

SRLW dst   dst = R, RX, IR, X

Operation:

```
tmp      ← dst
C        ← dst(0)
dst(15)  ← 0
dst(n)   ← tmp(n+1) for n = 0 to 14
```

The contents of the destination operand are shifted right one bit position. Bit 0 of the destination operand is moved to the Carry flag and zero is shifted into the most significant bit of the destination.

Flags:

S: Cleared
Z: Set if the result is zero; cleared otherwise
H: Cleared
P: Set if parity of the result is even; cleared otherwise
N: Cleared
C: Set if the bit shifted from bit 0 was a 1; cleared otherwise

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	SRLW R	11101101 11001011 001110rr	2	
RX:	SRLW RX	11101101 11001011 0011110y	2	
IR:	SRLW (HL)	11101101 11001011 00111010	2+r	
X:	SRLW (XY+d)	11y11101 11001011 ——d— 00111010	4+r	I

Field Encodings:
rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

SUB
SUBTRACT (BYTE)

SUB A,src src = R, RX, IM, IR, X

Operation: A ← A - src

The source operand is subtracted from the accumulator and the difference is stored in the accumulator. The contents of the source are unaffected. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 4 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the operands are of different signs and the result is of the same sign as the source; cleared otherwise
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SUB A,R	10010-r-	2	
RX:	SUB A,RX	11y11101 1001010w	2	
IM:	SUB A,n	11010110 ——n—	2	
IR:	SUB A,(HL)	10010110	2+r	
X:	SUB A,(XY+d)	11y11101 10010110 ——d—	4+r	I

Field Encodings:

r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

SUB
SUBTRACT (WORD)

SUB HL,src   src = DA

Operation:

```
if (XM) then begin
    HL(31-0)  ←  HL(31-0) - src(31-0)
    end
else begin
    HL(15-0)  ←  HL(15-0) - src(15-0)
    end
```

The source operand is subtracted from the HL register and the difference is stored in the HL register. The contents of the source are unaffected. Two's complement subtraction is performed. Note that the length of the operand is controlled by the Extended/Native mode selection, which is consistent with the manipulation of an address by the instruction.

Flags:

●

S: Unaffected
Z: Unaffected
H: Set if there is a borrow from bit 12 of the result; cleared otherwise
V: Unaffected
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
DA:	SUB HL,(nn)	11101101 11010110 -n(low)- -n(high)	2+r	I, X

SUB
SUBTRACT FROM STACK POINTER (WORD)

SUB SP,src   src = IM

Operation:

```
if (XM) then begin
    SP(31-0)   <-   SP(31-0) – src(31-0)
    end
else begin
    SP(15-0)   <-   SP(15-0) – src(15-0)
    end
```

The source operand is subtracted from the SP register and the difference is stored in the SP register. This has the effect of allocating or deallocating space on the stack. Two's complement subtraction is performed.

Flags:

S: Unaffected
Z: Unaffected
H: Set if there is a borrow from bit 12 of the result; cleared otherwise
V: Unaffected
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
IM:	SUB SP,nn	11101101 10010010 -n(low)- -n(high)	2	I, X

SUBW
SUBTRACT (WORD)

SUBW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) ← HL(15-0) - src(15-0)

The source operand is subtracted from the HL register and the difference is stored in the HL register. The contents of the source are unaffected. Two's complement subtraction is performed.

Flags:

S: Set if the result is negative; cleared otherwise
Z: Set if the result is zero; cleared otherwise
H: Set if there is a borrow from bit 12 of the result; cleared otherwise
V: Set if arithmetic overflow occurs, that is, if the operands are of different signs and the result is of the same sign as the source; cleared otherwise
N: Set
C: Set if there is a borrow from the most significant bit of the result; cleared otherwise

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SUBW [HL,]R	11101101 100101rr	2	
RX:	SUBW [HL,]RX	11y11101 10010111	2	
IM:	SUBW [HL,]nn	11101101 10010110 -n(low)- n(high)-	2	
X:	SUBW [HL,](XY+d)	11y11101 11010110 ——d—	2+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

SWAP
SWAP UPPER REGISTER WORD WITH LOWER REGISTER WORD

SWAP src src = R, RX

Operation: src(31-16) ↔ src(15-0)

The contents of the most significant word of the source are exchanged with the contents of the least significant word of the source.

Flags:
S: Unaffected
Z: Unaffected
H: Unaffected
V: Unaffected
N: Unaffected
C: Unaffected

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	SWAP R	11101101 00rr1110	2	
RX:	SWAP RX	11y11101 00111110	2	

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

TST
TEST (BYTE)

TST src src = R, IM, IR

Operation: A AND src

A logical AND operation is performed between the corresponding bits of the source operand and the accumulator. The contents of both the accumulator and the source are unaffected; only the flags are modified as a result of this instruction.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if all bits of the result are zero; cleared otherwise
H: Set
P: Set if the parity is even; cleared otherwise
N: Cleared
C: Cleared

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	TST R	11101101 00-r-100	2	
IM:	TST n	11101101 01100100 ——n—	2	
IR:	TST (HL)	11101101 00110100	2+r	

Field Encodings: r: per convention

TSTIO
TEST I/O PORT

TSTIO src src = IM

Operation: (C) AND src

A logical AND operation is performed between the corresponding bits of the source and the contents of the I/O location. The contents of both the I/O location and the source are unaffected; only the flags are modified as a result of this instruction. No external I/O transaction will be generated as a result of this instruction, although the I/O address will appear on the adress bus while the internal read is occurring. The peripheral address in the C register is placed on the low byte of the address bus and zeros are placed on all other address lines.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if all bits of the result are zero; cleared otherwise
H: Set
P: Set if the parity is even; cleared otherwise
N: Cleared
C: Cleared

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
	TSTIO n	11101101 01110100 ——n—	3+i	

XOR
EXCLUSIVE OR (BYTE)

XOR [A,]src src = R, RX, IM, IR, X

Operation: A ← A XOR src

A logical EXCLUSIVE OR operation is performed between the corresponding bits of the source operand and the accumulator and the result is stored in the accumulator. A 1 bit is stored wherever the corresponding bits in the two operands are different; otherwise a 0 bit is stored. The contents of the source are unaffected.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if all bits of the result are zero; cleared otherwise
H: Cleared
P: Set if the parity is even; cleared otherwise
N: Cleared
C: Cleared

Addressing Mode	Syntax	Instruction Format	Execute Time	Note
R:	XOR [A,]R	10101-r-	2	
RX:	XOR [A,]RX	11y11101 1010110w	2	
IM:	XOR [A,]n	11101110 ——n—	2	
IR:	XOR [A,](HL)	10101110	2+r	
X:	XOR [A,](XY+d)	11y11101 10101110 ——d—	4+r	I

Field Encodings:
r: per convention
y: 0 for IX, 1 for IY
w: 0 for high byte, 1 for low byte

XORW
EXCLUSIVE OR (WORD)

XORW [HL,]src src = R, RX, IM, X

Operation: HL(15-0) ← HL(15-0) XOR src(15-0)

A logical EXCLUSIVE OR operation is performed between the corresponding bits of the source operand and the HL register and the result is stored in the HL register. A 1 bit is stored wherever the corresponding bits in the two operands are different; otherwise a 0 bit is stored. The contents of the source are unaffected.

Flags:

S: Set if the most significant bit of the result is set; cleared otherwise
Z: Set if all bits of the result are zero; cleared otherwise
H: Cleared
P: Set if the parity is even; cleared otherwise
N: Cleared
C: Cleared

Addressing Mode	**Syntax**	**Instruction Format**	**Execute Time**	**Note**
R:	XORW [HL,]R	11101101 101011rr	2	
RX:	XORW [HL,]RX	11y11101 10101111	2	
IM:	XORW [HL,]nn	11101101 10101110 -n(low) -n(high)-	2	
X:	XORW [HL,](XY+d)	11y11101 11101110 ——d—	4+r	I

Field Encodings: rr: 00 for BC, 01 for DE, 11 for HL
y: 0 for IX, 1 for IY

ZiLOG

Table of Contents

Z380™ Architectural Overview 1

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

CHAPTER 6

INTERRUPTS AND TRAPS

6.1 INTRODUCTION

Exceptions are conditions that can alter the normal flow of program execution. The Z380™ CPU supports three kinds of exceptions; interrupts, traps, and resets.

Interrupts are asynchronous events generated by a device external to the CPU; peripheral devices use interrupts to request service from the CPU. Traps are synchronous events generated internally in the CPU by a particular condition that can occur during the attempted execution of an instruction—in particular, when executing undefined instructions. Thus, the difference between Traps and Interrupts is their origin. A Trap condition is always reproducible by re-executing the program that created the Trap, whereas an Interrupt is generally independent of the currently executing task.

A hardware reset overrides all other conditions, including Interrupts and Traps. It occurs when the /RESET line is activated and causes certain CPU control registers to be initialized. Resets are discussed in detail in Chapter 7.

The Z380 MPU's Interrupt and Trap structure provides compatibility with the existing Z80 and Z180 MPU's with the following exception—the undefined opcode Trap occurrence is with respect to the Z380 instruction set, and its response is improved (vs the Z180) to make Trap handling easier. The Z380 MPU also offers additional features to enhance flexibility in system design.

6.2 INTERRUPTS

Of the five external Interrupt inputs provided, one is assigned as a Nonmaskable Interrupt, /NMI. The remaining inputs, /INT3-/INTO, are four asynchronous maskable Interrupt requests.

The Nonmaskable Interrupt; (NMI) is an Interrupt that cannot be disabled (masked) by software. Typically NMI is reserved for high priority external events that need immediate attention, such as an imminent power failure. Maskable Interrupts are Interrupts that can be disabled (masked) through software by cleaning the appropriate bits in the Interrupt Enable Register (IER) and IEF1 bit in the Select Register (SR).

All of these four maskable Interrupt inputs (/INT3-/INT0) are external input signals to the Z380 CPU core. The four Interrupt enable bits in the Interrupt Enable Register determine (IER; Internal I/O address: 17H) which of the requested Interrupts are accepted. Each Interrupt input has a fixed priority, with /INT0 as the highest and /INT3 as the lowest.

The Enable Interrupt (EI) instruction is used to selectively enable the maskable Interrupts (by setting the appropriate bits in the IER register and IEF1 bit in the SR register) and the Disable Interrupt instruction is used to selectively disable interrupts (by clearing appropriate bits in the IER, and/or clearing IEF1 bit in the SR register). When an Interrupt source has been disabled, the CPU ignores any request from that source. Because maskable Interrupt requests are not retained by the CPU, the request signal on a maskable Interrupt line must be asserted until the CPU acknowledges the request.

When enabling Interrupts with the EI instruction, all maskable Interrupts are automatically disabled (whether previously enabled or not) for the duration of the execution of the EI instruction and the instruction immediately following.

Interrupts are always accepted between instructions. The block move, block search, and block I/O instructions can be interrupted after any iteration.

The Z380 CPU has four selectable modes for handling externally generated Interrupts, using the IM instruction. The first three modes extend the Z80 CPU Interrupt Modes to accommodate the Z380 CPU's additional Interrupt inputs in a compatible fashion. The fourth mode allows more flexibility in interrupt handling.

6.2 INTERRUPTS (Continued)

In an Interrupt acknowledge transaction, address outputs A31-A4 are driven to logic 1. One output among A3-A0 is driven to logic 0 to indicate the maskable interrupt request being acknowledged. If /INT0 is being acknowledged, A3-A1 are at logic 1 and A0 is at logic 0.

For the maskable Interrupt on /INT0 input, Interrupt Modes 0 through 3 are supported. Modes 0, 1, and 2 have the same schemes as those in the Z80 and Z180 MPU's. Mode 3 is similar to mode 2, except that 16-bit Interrupt vectors are expected from the I/O devices. Note that 8-bit and 16-bit I/O devices can be intermixed in this mode by having external pull-up resistors at the data bus signals D15-D8, for example.

The external maskable Interrupt requests /INT3-/INT1 are always handled in an assigned Interrupt vectors mode regardless of the current Interrupt Mode (IM3-IM0) in effect.

As discussed in the CPU Architecture section, the Z380 MPU can operate in either the Native or Extended mode. In Native mode, pushing and popping of the stack to save and retrieve interrupted PC values in Interrupt handling are done in 16-bit sizes, and the Stack Pointer rolls over at the 64 Kbyte boundary. In Extended mode, the PC pushes and pops are done in 32-bit sizes, and the Stack Pointer rolls over at the 4 Gbyte memory space boundary. The Z380 MPU provides an Interrupt Register Extension, whose contents are always output as the address bus signals A31-A16 when fetching the starting addresses of service routines from memory in Interrupt Modes 2, 3, and the assigned vectors mode. In Native mode, such fetches are automatically done in 16-bit sizes and in Extended mode, in 32-bit sizes. These starting addresses should be even-aligned in memory locations. That is, their least significant bytes should have addresses with A0 = 0.

6.2.1 Interrupt Priority Ranking

The Z380 MPU assigns a fixed priority ranking to handle its Interrupt sources, as shown in Table 6-1.

Table 6-1. Interrupt Priority Ranking

Priority	Interrupt Sources
Highest	Trap (undefined opcode)
	/NMI
	/INT0
	/INT1
	/INT2
Lowest	/INT3

6.2.2 Interrupt Control

The Z380 MPU's flags and registers associated with Interrupt processing are listed in Table 6-2. As discussed in the Chapter 1, "CPU Architecture," some of these registers reside in the on-chip I/O address space, and can be accessed only with reserved on-chip I/O instructions.

Table 6-2. Interrupt Flags and Registers

Names	Mnemonics	Access Methods
Interrupt Enable Flags	IEF1,IEF2	EI and DI Instructions
Interrupt Register	I	LD I,A and LD A,I Instructions
Interrupt Register Extension	Iz	LD I,HL and LD HL,I Instructions (Accessing both Iz and I)
Interrupt Enable Register	IER	On-chip I/O Instructions, Address 17H EI and DI Instruction
Assigned Vectors Base and Trap Register	AVBR	On-Chip I/O Instructions, Address 18H
Trap and Break Register	TRPBK	On-Chip I/O Instructions, Address 19H

6.2.2.1 IEF1, IEF2

IEF1 controls the overall enabling and disabling of all on-chip peripheral and external maskable Interrupt requests. If IEF1 is at logic 0, all such Interrupts are disabled. The purpose of IEF2 is to correctly manage the occurrence of /NMI. When /NMI is acknowledged, the state of IEF1 is copied to IEF2 and then IEF1 is cleared to logic 0. At the end of the /NMI interrupt service routine, execution of the Return From Nonmaskable Interrupt instruction, RETN, automatically copies the state of IEF2 back to IEF1. This is a means to restore the Interrupt enable condition existing before the occurrence of /NMI. Table 6-3 summarizes the states of IEF1 and IEF2 resulting from various operations.

Table 6-3. Operation Effects on IEF1 and IEF2

Operation	IEF1	IEF2	Comments
/RESET	0	0	Inhibits all interrupts except Trap and /NMI.
Trap	0	0	Disables interrupt nesting.
/NMI	0	IEF1	IEF1 value copied to IEF2, then IEF1 is cleared.
RETN	IEF2	NC	Returns from /NMI service routine.
/INT3-/INT0	0	0	Disables interrupt nesting.
RETI	NC	NC	Returns from Interrupt service routine, Z80 I/O device.
RET	NC	NC	Returns from service routine, or returns from Interrupt service routine for a non-Z80 I/O device.
EI	1	1	
DI	0	0	
LD A,I or LD R,I	NC	NC	IEF2 value is copied to P/V Flag.
LD HL,I or LD HL,R	NC	NC	

(NC = No Change)

6.2.2.2 I, I Extend

The 8-bit Interrupt Register and the 16-bit Interrupt Register Extension are cleared during reset.

6.2.2.3 Interrupt Enable Register

D7-D4 Reserved Read as 0, should write to as 0.
D3-D0 IE3-IE0 (Interrupt Request Enable Flags)

These flags individually indicate if /INT3, /INT2, /INT1, or /INT0 is enabled. Note that these flags are conditioned with the Enable and Disable Interrupt instructions (with arguments) (See Figure 6.1).

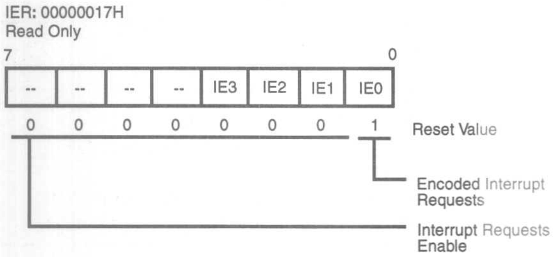

IER: 00000017H
Read Only
7
0
--
--
--
--
IE3
IE2
IE1
IE0
0 0 0 0 0 0 0 1
Reset Value
Encoded Interrupt Requests
Interrupt Requests Enable

Figure 6-1. Interrupt Enable Register

6.2.2.4 Assigned Vectors Base Register

D7-D1 AB15-AB9 (Assigned Vectors Base). The Interrupt Register Extension, Iz, together with AB15-AB9, define the base address of the assigned Interrupt vectors table in memory space (See Figure 6-2).

D0 Reserved. Read as 0, should write to as 0.

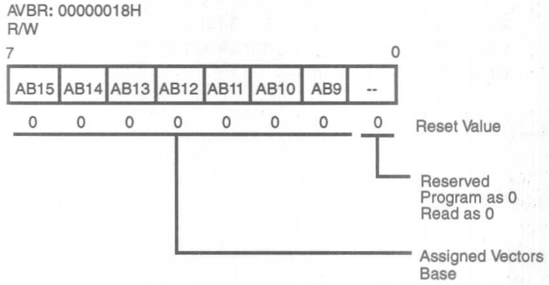

AVBR: 00000018H
R/W
7
0
AB15
AB14
AB13
AB12
AB11
AB10
AB9
--
0 0 0 0 0 0 0 0
Reset Value
Reserved
Program as 0
Read as 0
Assigned Vectors Base

Figure 6-2. Assigned Vectors Base Register

6.2.2.5 Trap and Break Register

D7-D2 Reserved. Some of these bits are reserved for development support functions. Read as 0, should write to as 0.

D1 TF (Trap on Instruction Fetch). TF goes active to logic 1 when an undefined opcode fetched in the instruction stream is detected. TF can be reset under program control by writing it with a logic 0. However, it cannot be written with a logic 1.

D0 TV (Trap on Interrupt Vector). TV goes active to logic 1 when an undefined opcode is returned as a vector in an Interrupt acknowledge transaction in mode 0. TV can be reset under program control by writing it with a logic 0. However, it cannot be written with a logic 1 (See Figure 6-3).

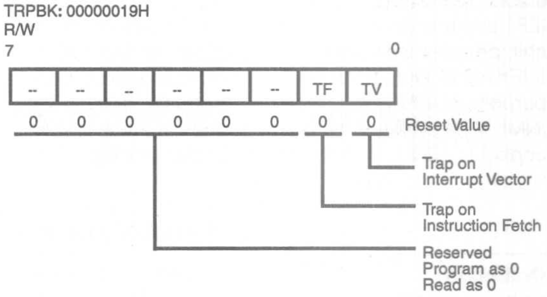


Figure 6-3. Trap and Break Register

6.3 TRAP INTERRUPT

The Z380 MPU generates a Trap when an undefined opcode is encountered. The Trap is enabled immediately after reset, and it is not maskable. This feature can be used to increase software reliability or to implement "extended" instructions. An undefined opcode can be fetched from the instruction stream, or it can be returned as a vector in an Interrupt acknowledge transaction in Interrupt Mode 0. When a Trap occurs, the Z380 MPU operates as follows.

1. The TF or TV bit in the Assigned Vectors Base and Trap Register goes active, to indicate the source of the undefined opcode.

2. If the undefined opcode was fetched from the instruction stream, the starting address of the Trap causing the instruction is pushed onto the stack. (Note that the starting address of decoder directive(s) preceding an instruction encoding is considered the starting address of the instruction.)

   If the undefined opcode was a returned Interrupt vector, the interrupted PC value is pushed onto the stack.

3. The states of IEF1 and IEF2 are cleared.

4. The Z380 MPU commences to fetch and execute instructions from address 00000000H.

Note that instruction execution resumes at address 0, similar to the occurrence of a reset. Testing the TF and TV bits in the Assigned Vectors Base and Trap Register will distinguish the two events. Even if Trap handling is not in place, repeated restarts from address 0 is an indicator of possible illegal instructions at system debugging.

6.4 NONMASKABLE INTERRUPT

The Nonmaskable Interrupt Input /NMI is edge sensitive, with the Z380 MPU internally latching the occurrence of its falling edge. When the latched version of /NMI is recognized, the following operations are performed.

1. The Interrupted PC (Program Counter) value is pushed onto the stack. The size of the PC value pushed onto the stack depends on Native (one word) or Extended mode (two words) in effect.
2. The state of IEF1 is copied to IEF2, then IEF1 is cleared.
3. The Z380 MPU commences to fetch and execute instructions from address 00000066H.

6.5 INTERRUPT RESPONSE FOR MASKABLE INTERRUPT ON /INT0

The transactions caused by the Maskable Interrupt on /INT0 are different depends on the Interrupt Mode in effect at the time when the interrupt has been accepted, as described below.

6.5.1 Interrupt Mode 0 Response for Maskable Interrupt /INT0

This mode is similar to the 8080 CPU Interrupt response mode. During the Interrupt acknowledge transaction, the external I/O device being acknowledged is expected to output a vector onto the upper portion of the data bus, D15-D8. The Z380 MPU interprets the vector as an instruction opcode. IEF1 and IEF2 are reset to logic 0, disabling all further maskable interrupt requests. Note that unlike the other interrupt responses, the PC is not automatically pushed onto the stack. Typically, a Restart instruction (RST) is used, since the Restart opcode is only one byte long, meaning that the interrupting peripheral needs to supply only one byte of information. For this case, it pushes the interrupted PC (Program Counter) value onto the stack and resumes execution at a fixed memory location. Alternatively, a 3-byte call to any location can be executed.

Note that a Trap occurs if an undefined opcode is supplied by the I/O device as a vector.

6.5.2 Interrupt Mode 1 Response for Maskable Interrupt /INT0

In Interrupt Mode 1, the Z380 CPU automatically executes a Restart to a fixed location (00000038H) when an interrupt occurs. An Interrupt acknowledge transaction is generated, during which the data bus contents are ignored by the Z380 MPU. The interrupted PC value is pushed onto the stack. The size of the PC value pushed onto the stack is depends on Native (one word) or Extended mode (two words) in effect. The IEF1 and IEF2 are reset to logic 0 so as to disable further maskable interrupt requests. Instruction fetching and execution restarts at memory location 00000038H.

6.5.3 Interrupt Mode 2 Response for Maskable Interrupt /INT0

Interrupt Mode 2 is a vectored Interrupt response mode, wherein the interrupting device identifies the starting location of service routine using an 8-bit vector read by the CPU during the Interrupt acknowledge cycle.

During the Interrupt acknowledge transaction, the external I/O device being acknowledged is expected to output a vector onto the upper portion of the data bus, D15-D8. The interrupted PC value is pushed onto the stack and IEF1 and IEF2 are reset to logic 0 so as to disable further maskable interrupt requests. The size of the PC value pushed onto the stack is depends on Native (one word) or Extended mode (two words) in effect. The Z380 MPU then reads an entry from a table residing in memory and loads it into the PC to resume execution. The address of the table entry is composed of the I Extend (Iz) contents as A31-A16, the I Register contents as A15-A8 and the vector supplied by the I/O device as A7-A0. Note that the table entry is effectively the starting address of the interrupt service routine designed for the I/O device being acknowledged, and the table composing of starting addresses for all the Interrupt Mode 2 service routines can be referred to as the Interrupt Mode 2 vector table. Each table entry should be word-sized if the Z380 MPU is in the Native mode and Long Word-sized if in the Extended mode, in either case even-aligned (least significant byte with address A0 = 0), meaning 128 different vectors can be used in the Native mode, and 64 different vectors can be used in Extended mode.

6.5.4 Interrupt Mode 3 Response for Maskable Interrupt /INT0

Interrupt Mode 3 is similar to mode 2 except that a 16-bit vector is expected to be placed on the data bus D15-D0 by the I/O device during the Interrupt acknowledge transaction. The interrupted PC is pushed onto the stack. The size of the PC value pushed onto the stack depends on the

6.5.4 Interrupt Mode 3 Response for Maskable Interrupt /INT0 (Continued)

Native (one word) or Extended mode (two words) in effect. IEF1 and IEF2 are reset to logic 0 so as to disable further maskable Interrupt requests. The starting address of the service routine is fetched and loaded into the PC to resume execution, from memory location with an address composed of the I Extend contents as A31-A16 and the vector supplied by the I/O device as A15-A0. Again the starting address of the service routine is word-sized if the Z380 MPU is in Native mode and Long Word-sized if in the Extended mode, in either case even-aligned, meaning 32768 different vectors can be used in the Native mode, and 16384 different vectors can be used in the Extended mode.

6.6 ASSIGNED INTERRUPT VECTORS MODE FOR MASKABLE INTERRUPTS /INT3-/INT1

Regardless of the Interrupt Mode in effect, interrupts on /INT3-/INT1 is always handled by the Assigned Interrupt Mode. This mode is similar to the interrupt handling on the Z180's /INT1 or /INT2 line. When the Z380 MPU recognizes one of the external maskable Interrupts /INT3-/INT1, it generates an Interrupt acknowledge transaction which is different than that for /INT0. The Interrupt acknowledge transaction for /INT3-/INT1 has the I/O bus signal /INTACK active, with /M1 /IORQ, /IORD, and /IOWR inactive. The interrupted PC value is pushed onto the stack. The size of the PC value pushed onto the stack is depends on the Native (one word) or Extended mode (two words) in effect. IEF1 and IEF2 are reset to logic 0, disabling further maskable Interrupt requests. The starting address of an Interrupt service routine is fetched from a table entry and loaded into the PC to resume execution. The address of the table entry is composed of the I Extend contents as A31-A16, the AB bits of the Assigned Vectors Base Register as A15-A9, and an assigned interrupt vector specific to the request being recognized as A8-A0. The assigned vectors are defined in Table 6-4. If the Z380 CPU is in Extended mode, all four bytes of the data stored in the Assigned vector location will be used as a new PC value. If the Z380 CPU is in Native mode, only two bytes of data from the LS Byte will be used as a new PC value.

Table 6-4. Assigned Interrupt Vectors

Interrupt Source	Assigned Interrupt Vector
/INT1	00H
/INT2	04H
/INT3	08H

6.7 RETI INSTRUCTION

The Z80 family I/O devices are designed to monitor the Return from Interrupt opcodes in the instruction stream (RETI — EDH, 4DH), signifying the end of the current Interrupt service routine. When detected, the daisy chain within and among the device(s) resolves and the appropriate Interrupt-under-service condition clears. The Z380 MPU "reproduces" the opcode fetch transactions on the I/O bus when the RETI instruction is executed. Note that the Z380 MPU outputs the RETI opcodes onto both portions of the data bus (D15-D8 and D7-D0) in the transactions.

Table of Contents

Z380™ Architectural Overview 1

Address Spaces 2

Mode of Operations and Decoder Directives 3

Addressing Modes and Data Types 4

Instruction Set 5

Interrupts and Traps 6

Reset 7

CHAPTER 7
RESET

7.1 INTRODUCTION

The Z380 CPU is placed in a dormant state when the /RESET input is asserted. All its operations are terminated, including any interrupt, bus request, or bus transaction that may be in progress. On the Z380 MPU, the IOCLK goes Low on the next BUSCLK rising edge and enters into the BUSCLK divided-by-eight mode. The address and data buses are tri-stated, and the bus control signals are driven to their inactive states. The effect of /RESET on the Z380 CPU and related internal I/O registers is depicted in Table 7-1.

The /RESET input may be asynchronous to BUSCLK, though it is sampled internally at BUSCLK's falling edges. For proper initialization of the Z380 CPU, V_{DD} must be within operating specifications and the CLK input must be stable for more than five cycles with /RESET held Low.

The Z380 CPU proceeds to fetch the first instruction 3.5 BUSCLK cycles after /RESET is deasserted, provided such deassertion meets the proper setup and hold times with reference to the falling edge of BUSCLK. On the Z380 MPU implementation, with the proper setup and hold times being met, IOCLK's first rising edge is 11.5 BUSCLK cycles after the /RESET deassertion, preceded by a minimum of four BUSCLK cycles when IOCLK is at Low.

Note that if /BREQ is active when /RESET is deasserted, the Z380 MPU would relinquish the bus instead of fetching its first instruction. IOCLK synchronization would still take place as described before.

Requirements to reset the device, and the initial state after reset might be different depending on the particular implementation of the Z380 CPU on the individual Superintegration version of the device. For /RESET effects and requirements, refer to the individual product specification.

Table 7-1. Effect of a Reset on Z380 CPU and Related I/O Registers

Register	Reset Value	Comments
Program Counter	00000000	PCz, PC
Stack Pointer	00000000	SPz, SP
I R	000000 00	Iz, I
Select Register	00000000	Register Bank 0 Selected: AF, Main Bank, IX, IY Native Mode Maskable Interrupts Disabled, in Mode 0 Bus Request Lock-Off
A and F Registers		Register Banks 3-0: A, F, A', F' Unaffected
Register Extensions	0000	Register Bank 0: BCz, DEz, HLz, IYz, BCz', DEz', HLz', IYz' (All "non-extended" portions unaffected.) Register Bank 3-1 Unaffected.
I/O Bus Control Register 0	00	IOCLK = BUSCLK/8
Interrupt Enable Register	01	/INT0 Enabled
Assigned Vector Base Register	00	
Trap and Break Register	00	

Appendix A A

Appendix B B

Appendix C C

Appendix D D

Appendix E E

Index I

Superintegration™ Products Guide S

Zilog's Literature Guide Ordering Information L

APPENDIX A

Z380™ CPU INSTRUCTION FORMATS

Four formats are used to generate the machine language bit encoding for the Z380 CPU instructions. Also, the Z380 CPU has eight Decoder Directives which work as a special escape sequence to the certain instructions, to expand its capability as explained in Chapter 3.

The bit encoding of the Z380 CPU instructions are partitioned into bytes. Every instructions encoding contains one byte dedicated to specifying the type of operation to be performed; this byte is referred to as the instruction's operation code, or opcode. Besides specifying a particular operation, opcode typically include bit encoding specifying the operand addressing mode for the instruction and identifying any general purpose registers used by the instruction. Along with the opcode, instruction encoding may include bytes that contain an address, displacement, and/or immediate value used by the instruction, and special bytes called "escape codes" that determine the meaning of the opcode itself.

By themselves, one byte opcode would allow the encoding of only 256 unique instructions. Therefore, special "escape codes" that precede the opcode in the instruction encoding are used to expand the number of possible instructions. There are two types of escape codes; addressing mode and opcode. Escape codes for the Z80 original instructions are one bytes in length, and the escape codes used to expand the Z380 instructions are one or two bytes in length.

These instruction formats are differentiated by the opcode escape value used. Format 1 is for instructions without an opcode escape byte(s), Format 2 is for instructions with an opcode escape byte. Format 3 is for instructions whose opcode escape byte has the value 0CBH, and Format 4 is for instructions whose escape bytes are 0ED, followed by 0CBH.

For the opcode escape byte, the Z380 CPU uses 0DDH and 0FDH as well, which on the Z80 CPU, these are used only as an address escape byte.

In Format 2 and 4, the opcode escape byte immediately precedes the opcode byte itself.

In Format 3, a 1-byte displacement may be between the opcode escape byte and opcode itself. Opcode escape bytes are used to distinguish between two different instructions with the same opcode bytes, thereby allowing more than 256 unique instructions. For example, the 01H opcode, when alone, specifies a form of a Load Register Word instruction; when proceeded by 0CBH escape code, the opcode 01H specifies a Rotate Left Circular instruction.

Format 3 instructions with DDIR Immediate data Decoder Directives, 1 to 3 bytes of displacement is between the opcode escape byte and opcode itself.

Format 4 instructions are proceeded by 0EDH, 0CBH, and a opcode. Optionally, with immediate word field follows.

Addressing mode escape codes are used to determine the type of encoding for the addressing mode field within an instruction's opcode, and can be used in instructions with and without opcode escape value. An addressing mode escape byte can have the value of 0DDH or 0FDH. The addressing mode escape byte, if present, is always the first byte of the instruction's machine code, and is immediately followed by either the opcode (Format 1), or the opcode escape byte (Format 2 and 3). For example, the 46H opcode, when alone, specifies a Load B register from memory location pointed by (HL) register; when proceeded by the 0DDH escape byte, the opcode 46H specifies a Load B register from the memory location pointed by (IX+d).

The four instruction formats are shown in Tables A-1 through A-4. Within each format, several different configurations are possible, depending on whether the instruction involves addressing mode escape bytes, addresses, displacements, or immediate data. In Table A-1 through A-4, the symbol "A.esc" is used to indicate the presence of an addressing mode escape byte, "O.esc" is used to indicate the presence of an opcode escape byte, "disp." is an abbreviation for displacement and "addr." is an abbreviation for address.

Table A-1. Format 1 Instructions Encodings

	Instruction Format		Assembly	Hexadecimal
	Opcode		LD A,C	79
	Opcode	2-byte Address	LD A,(addr)	3A addr (L) addr (H)
	Opcode	1-byte Displacement	DJNZ addr	10 disp
	Opcode	Immediate	LD E,n	1E n
A.esc	Opcode	2-byte Address	LD IX,(addr)	DD 2A addr (L) addr (H)
A.esc	Opcode	1-byte Displacement	LD A, (IX+d)	DD 7E disp
A.esc	Opcode	Immediate	LD IX,nn	DD 21 n(L) n(H)
A.esc	Opcode	1-byte Displacement Immediate	LD (IY+d),n	FD 36 d n

Note: "A.esc" is an addressing mode escape byte, and either 0DDH or 0FDH.

Table A-2. Format 2 Instructions Encodings

	Instruction Format		Assembly	Hexadecimal
	Opcode		LD A,C	79
O.esc	Opcode	Immediate (1 byte)	TST n	ED 64 n
O.esc	Opcode	Immediate (2 bytes)	LD (BC),nn	ED 06 n(L) n(H)
O.esc	Opcode	Address (2 bytes)	LD BC,(addr)	ED 4B addr (L) addr (H)
O.esc	Opcode	Displacement (1 byte)	CALR e	ED CD e
O.esc	Opcode	Displacement (2 bytes)	JR ee	DD 18 d(L) d(H)
O.esc	Opcode	Displacement (3 bytes)	JR eee	FD 18 d(L) d(M) d(H)

Note: "O.esc" is an opcode escape byte, and either 0DDH, 0EDH or 0FDH.

Table A-3. Format 3 Instruction Encoding

	CB	Opcode	RLC (HL)	CB 06
A.esc	CB	1 Byte Displacement Opcode	RLC (IX+d)	DD CB d 06

Note: "A.esc" is an addressing mode escape byte, and either 0DDH or 0FDH.

Table A-4. Format 4 Instruction Encoding

ED	CB	Opcode	RRCW BC	ED CB 08
ED	CB	Opcode Immediate	MULTW nn	ED CB 97 n(L) n(H)

ZILOG

Appendix A A

Appendix B B

Appendix C C

Appendix D D

Appendix E E

Index I

Superintegration™ Products Guide S

Zilog's Literature Guide Ordering Information L

APPENDIX B

Z380™ INSTRUCTIONS IN ALPHABETIC ORDER

This Appendix contains a quick reference guide when programming.

It has the Z380 instructions sorted by alphabetic order.

The column "Mode" indicates whether the instruction is affected by DDIR immediate Decoder Directives, Extended mode or Native mode of operation, and Word or Long Word mode of operation; "I" means the instruction can be used with DDIR IM to expand its immediate constant, "X" means that the operation of the instruction is affected by the XM status bit, and "L" means that the instruction is affected by LW status bit, or can be used with DDIR LW or DDIR W. The Native/Extended modes, Word/Long Word modes and Decoder Directives are discussed in Chapter 3 in this manual.

Source Code		Mode	Object Code			
ADC	A,(HL)		8E			
ADC	A,(IX+12H)	I	DD	8E	12	
ADC	A,(IY+12H)	I	FD	8E	12	
ADC	A,A		8F			
ADC	A,B		88			
ADC	A,C		89			
ADC	A,D		8A			
ADC	A,E		8B			
ADC	A,H		8C			
ADC	A,IXL		DD	8D		
ADC	A,IXU		DD	8C		
ADC	A,IYL		FD	8D		
ADC	A,IYU		FD	8C		
ADC	A,L		8D			
ADC	HL,BC		ED	4A		
ADC	HL,DE		ED	5A		
ADC	HL,HL		ED	6A		
ADC	HL,SP		ED	7A		
ADCW	(IX+12H)	I	DD	CE	12	
ADCW	(IY+12H)	I	FD	CE	12	
ADCW	1234H		ED	8E	34	12
ADCW	BC		ED	8C		
ADCW	DE		ED	8D		
ADCW	HL		ED	8F		
ADCW	HL,(IX+12H)	I	DD	CE	12	
ADCW	HL,(IY+12H)	I	FD	CE	12	
ADCW	HL,1234H		ED	8E	34	12
ADCW	HL,BC		ED	8C		
ADCW	HL,DE		ED	8D		
ADCW	HL,HL		ED	8F		
ADCW	HL,IX		DD	8F		
ADCW	HL,IY		FD	8F		
ADCW	IX		DD	8F		
ADCW	IY		FD	8F		
ADD	A,(HL)		86			
ADD	A,(IX+12H)	I	DD	86	12	
ADD	A,(IY+12H)	I	FD	86	12	
ADD	A,12H		C6	12		
ADD	A,12H		CE	12		
ADD	A,A		87			
ADD	A,B		80			
ADD	A,C		81			
ADD	A,D		82			
ADD	A,E		83			
ADD	A,H		84			
ADD	A,IXL		DD	85		
ADD	A,IXU		DD	84		
ADD	A,IYL		FD	85		
ADD	A,IYU		FD	84		
ADD	A,L		85			
ADD	HL,(1234H)	I X	ED	C6	34	12
ADD	HL,BC	X	09			
ADD	HL,DE	X	19			
ADD	HL,HL	X	29			
ADD	HL,SP	X	39			
ADD	IX,BC	X	DD	09		
ADD	IX,DE	X	DD	19		
ADD	IX,IX	X	DD	29		
ADD	IX,SP	X	DD	39		
ADD	IY,BC	X	FD	09		
ADD	IY,DE	X	FD	19		
ADD	IY,IY	X	FD	29		
ADD	IY,SP	X	FD	39		
ADD	SP,1234H	I X	ED	82	34	12
ADDW	(IX+12H)	I	DD	C6	12	
ADDW	(IY+12H)	I	FD	C6	12	
ADDW	1234H		ED	86	34	12
ADDW	BC		ED	84		
ADDW	DE		ED	85		
ADDW	HL		ED	87		
ADDW	HL,(IX+12H)	I	DD	C6	12	
ADDW	HL,(IY+12H)	I	FD	C6	12	
ADDW	HL,1234H		ED	86	34	12
ADDW	HL,BC		ED	84		
ADDW	HL,DE		ED	85		
ADDW	HL,HL		ED	87		
ADDW	HL,IX		DD	87		
ADDW	HL,IY		FD	87		
ADDW	IX		DD	87		
ADDW	IY		FD	87		
AND	(HL)		A6			
AND	(IX+12H)	I	DD	A6	12	
AND	(IY+12H)	I	FD	A6	12	
AND	12H		E6	12		
AND	A		A7			
AND	A,(HL)		A6			
AND	A,(IX+12H)	I	DD	A6	12	
AND	A,(IY+12H)	I	FD	A6	12	
AND	A,12H		E6	12		
AND	A,A		A7			
AND	A,B		A0			
AND	A,C		A1			
AND	A,D		A2			
AND	A,E		A3			
AND	A,H		A4			
AND	A,IXL		DD	A5		
AND	A,IXU		DD	A4		
AND	A,IYL		FD	A5		
AND	A,IYU		FD	A4		
AND	A,L		A5			
AND	B		A0			
AND	C		A1			
AND	D		A2			
AND	E		A3			
AND	H		A4			
AND	IXL		DD	A5		
AND	IXU		DD	A4		
AND	IYL		FD	A5		

Source Code		Mode	Object Code			
AND	IYU		FD	A4		
AND	L		A5			
ANDW	(IX+12H)	I	DD	E6	12	
ANDW	(IY+12H)	I	FD	E6	12	
ANDW	1234H		ED	A6	34	12
ANDW	BC		ED	A4		
ANDW	DE		ED	A5		
ANDW	HL		ED	A7		
ANDW	HL,(IX+12H)	I	DD	E6	12	
ANDW	HL,(IY+12H)	I	FD	E6	12	
ANDW	HL,1234H		ED	A6	34	12
ANDW	HL,BC		ED	A4		
ANDW	HL,DE		ED	A5		
ANDW	HL,HL		ED	A7		
ANDW	HL,IX		DD	A7		
ANDW	HL,IY		FD	A7		
ANDW	IX		DD	A7		
ANDW	IY		FD	A7		
BIT	0,(HL)		CB	46		
BIT	0,(IX+12H)	I	DD	CB	12	46
BIT	0,(IY+12H)	I	FD	CB	12	46
BIT	0,A		CB	47		
BIT	0,B		CB	40		
BIT	0,C		CB	41		
BIT	0,D		CB	42		
BIT	0,E		CB	43		
BIT	0,H		CB	44		
BIT	0,L		CB	45		
BIT	1,(HL)		CB	4E		
BIT	1,(IX+12H)	I	DD	CB	12	4E
BIT	1,(IY+12H)	I	FD	CB	12	4E
BIT	1,A		CB	4F		
BIT	1,B		CB	48		
BIT	1,C		CB	49		
BIT	1,D		CB	4A		
BIT	1,E		CB	4B		
BIT	1,H		CB	4C		
BIT	1,L		CB	4D		
BIT	2,(HL)		CB	56		
BIT	2,(IX+12H)	I	DD	CB	12	56
BIT	2,(IY+12H)	I	FD	CB	12	56
BIT	2,A		CB	57		
BIT	2,B		CB	50		
BIT	2,C		CB	51		
BIT	2,D		CB	52		
BIT	2,E		CB	53		
BIT	2,H		CB	54		
BIT	2,L		CB	55		
BIT	3,(HL)		CB	5E		
BIT	3,(IX+12H)	I	DD	CB	12	5E
BIT	3,(IY+12H)	I	FD	CB	12	5E
BIT	3,A		CB	5F		
BIT	3,B		CB	58		
BIT	3,C		CB	59		

Source Code		Mode		Object Code			
BIT	3,D			CB	5A		
BIT	3,E			CB	5B		
BIT	3,H			CB	5C		
BIT	3,L			CB	5D		
BIT	4,(HL)			CB	66		
BIT	4,(IX+12H)	I		DD	CB	12	66
BIT	4,(IY+12H)	I		FD	CB	12	66
BIT	4,A			CB	67		
BIT	4,B			CB	60		
BIT	4,C			CB	61		
BIT	4,D			CB	62		
BIT	4,E			CB	63		
BIT	4,H			CB	64		
BIT	4,L			CB	65		
BIT	5,(HL)			CB	6E		
BIT	5,(IX+12H)	I		DD	CB	12	6E
BIT	5,(IY+12H)	I		FD	CB	12	6E
BIT	5,A			CB	6F		
BIT	5,B			CB	68		
BIT	5,C			CB	69		
BIT	5,D			CB	6A		
BIT	5,E			CB	6B		
BIT	5,H			CB	6C		
BIT	5,L			CB	6D		
BIT	6,(HL)			CB	76		
BIT	6,(IX+12H)	I		DD	CB	12	76
BIT	6,(IY+12H)	I		FD	CB	12	76
BIT	6,A			CB	77		
BIT	6,B			CB	70		
BIT	6,C			CB	71		
BIT	6,D			CB	72		
BIT	6,E			CB	73		
BIT	6,H			CB	74		
BIT	6,L			CB	75		
BIT	7,(HL)			CB	7E		
BIT	7,(IX+12H)	I		DD	CB	12	7E
BIT	7,(IY+12H)	I		FD	CB	12	7E
BIT	7,A			CB	7F		
BIT	7,B			CB	78		
BIT	7,C			CB	79		
BIT	7,D			CB	7A		
BIT	7,E			CB	7B		
BIT	7,H			CB	7C		
BIT	7,L			CB	7D		
BTEST				ED	CF		
CALL	1234H	I	X	CD	34	12	
CALL	C,1234H	I	X	DC	34	12	
CALL	M,1234H	I	X	FC	34	12	
CALL	NC,1234H	I	X	D4	34	12	
CALL	NZ,1234H	I	X	C4	34	12	
CALL	P,1234H	I	X	F4	34	12	
CALL	PE,1234H	I	X	EC	34	12	
CALL	V, 1234H	I	X	EC	34	12	
CALL	PO,1234H	I	X	E4	34	12	

Source Code		Mode		Object Code				
CALL	NV, 1234H	I	X	E4	34	12		
CALL	Z,1234H	I	X	CC	34	12		
CALR	123456H		X	FD	CD	56	34	12
CALR	1234H		X	DD	CD	34	12	
CALR	12H		X	ED	CD	12		
CALR	C,123456H		X	FD	DC	56	34	12
CALR	C,1234H		X	DD	DC	34	12	
CALR	C,12H		X	ED	DC	12		
CALR	M,123456H		X	FD	FC			
CALR	M,1234H		X	DD	FC	34	12	
CALR	M,12H		X	ED	FC	12		
CALR	NC,123456H		X	FD	D4	56	34	12
CALR	NC,1234H		X	DD	D4	34	12	
CALR	NC,12H		X	ED	D4	12		
CALR	NZ,123456H		X	FD	C4	56	34	12
CALR	NZ,1234H		X	DD	C4	34	12	
CALR	NZ,12H		X	ED	C4	12		
CALR	P,123456H		X	FD	F4	56	34	12
CALR	P,1234H		X	DD	F4	34	12	
CALR	P,12H		X	ED	F4	12		
CALR	PE,123456H		X	FD	EC	56	34	12
CALR	PE,1234H		X	DD	EC	34	12	
CALR	PE,12H		X	ED	EC	12		
CALR	PO,123456H		X	FD	E4	56	34	12
CALR	PO,1234H		X	DD	E4	34	12	
CALR	PO,12H		X	ED	E4	12		
CALR	Z,123456H		X	FD	CC	56	34	12
CALR	Z,1234H		X	DD	CC	34	12	
CALR	Z,12H		X	ED	CC	12		
CCF				3F				
CP	(HL)			BE				
CP	(IX+12H)	I		DD	BE	12		
CP	(IY+12H)	I		FD	BE	12		
CP	12H			FE	12			
CP	A			BF				
CP	A,(HL)			BE				
CP	A,(IX+12H)	I		DD	BE	12		
CP	A,(IY+12H)	I		FD	BE	12		
CP	A,12H			FE	12			
CP	A,A			BF				
CP	A,B			B8				
CP	A,C			B9				
CP	A,D			BA				
CP	A,E			BB				
CP	A,H			BC				
CP	A,IXL			DD	BD			
CP	A,IXU			DD	BC			
CP	A,IYL			FD	BD			
CP	A,IYU			FD	BC			
CP	A,L			BD				
CP	B			B8				
CP	C			B9				
CP	D			BA				
CP	E			BB				

Source Code		Mode		Object Code			
CP	H			BC			
CPW	HL,IX			DD	BF		
CPW	IX			DD	BF		
CP	IXL			DD	BD		
CP	IXU			DD	BC		
CP	IYL			FD	BD		
CP	IYU			FD	BC		
CP	L			BD			
CPD			X	ED	A9		
CPDR			X	ED	B9		
CPI			X	ED	A1		
CPIR			X	ED	B1		
CPL	A			2F			
CPL				2F			
CPLW	HL			DD	2F		
CPLW				DD	2F		
CPW	(IX+12H)	I		DD	FE	12	
CPW	(IY+12H)	I		FD	FE	12	
CPW	1234H			ED	BE	34	12
CPW	BC			ED	BC		
CPW	DE			ED	BD		
CPW	HL			ED	BF		
CPW	HL,(IX+12H)	I		DD	FE	12	
CPW	HL,(IY+12H)	I		FD	FE	12	
CPW	HL,1234H			ED	BE	34	12
CPW	HL,BC			ED	BC		
CPW	HL,DE			ED	BD		
CPW	HL,HL			ED	BF		
CPW	HL,IY			FD	BF		
CPW	IY			FD	BF		
DAA				27			
DDIR	IB			DD	C3		
DDIR	IB,LW			FD	C1		
DDIR	IB,W			DD	C1		
DDIR	IW			FD	C3		
DDIR	IW,LW			FD	C2		
DDIR	IW,W			DD	C2		
DDIR	LW			FD	C0		
DDIR	W			DD	C0		
DEC	(HL)			35			
DEC	(IX+12H)	I		DD	35	12	
DEC	(IY+12H)	I		FD	35	12	
DEC	A			3D			
DEC	B			05			
DEC	BC		X	0B			
DEC	C			0D			
DEC	D			15			
DEC	DE		X	1B			
DEC	E			1D			
DEC	H			25			
DEC	HL		X	2B			
DEC	IX		X	DD	2B		
DEC	IXL			DD	2D		
DEC	IXU			DD	25		

Source Code		Mode	Object Code
DEC	IY	X	FD 2B
DEC	IYL		FD 2D
DEC	IYU		FD 25
DEC	L		2D
DEC	SP	X	3B
DECW	BC	X	0B
DECW	DE	X	1B
DECW	HL	X	2B
DECW	IX	X	DD 2B
DECW	IY	X	FD 2B
DECW	SP	X	3B
DI	1FH		DD F3 1F
DI			F3
DIVUW	(IX+12H)	I	DD CB 12 BA
DIVUW	(IY+12H)	I	FD CB 12 BA
DIVUW	1234H		ED CB BF
DIVUW	BC		ED CB B8
DIVUW	DE		ED CB B9
DIVUW	HL		ED CB BB
DIVUW	HL,(IX+12H)	I	DD CB 12 BA
DIVUW	HL,(IY+12H)	I	FD CB 12 BA
DIVUW	HL,1234H		ED CB BF
DIVUW	HL,BC		ED CB B8
DIVUW	HL,DE		ED CB B9
DIVUW	HL,HL		ED CB BB
DIVUW	HL,IX		ED CB BC
DIVUW	HL,IY		ED CB BD
DIVUW	IX		ED CB BC
DIVUW	IY		ED CB BD
DJNZ	123456H	X	FD 10 56 34 12
DJNZ	1234H	X	DD 10 34 12
DJNZ	12H	X	10 12
EI	1FH		DD FB 1F
EI			FB
escape			CB
escape			DD
escape			ED
escape			FD
escape			ED CB
escape			DD CB
escape			FD CB
EX	(SP),HL	L	E3
EX	(SP),IX	L	DD E3
EX	(SP),IY	L	FD E3
EX	A,(HL)		ED 37
EX	A,A		ED 3F
EX	A,A'		CB 37
EX	A,B		ED 07
EX	A,C		ED 0F
EX	A,D		ED 17
EX	A,E		ED 1F
EX	A,H		ED 27
EX	A,L		ED 2F
EX	AF,AF'		08
EX	B,B'		CB 30
EX	BC,BC'	L	ED CB 30
EX	BC,DE	L	ED 05
EX	BC,HL	L	ED 0D
EX	BC,IX	L	ED 03
EX	BC,IY	L	ED 0B
EX	C,C'		CB 31
EX	D,D'		CB 32
EX	DE,DE'	L	ED CB 31
EX	DE,HL	L	EB
EX	DE,IX	L	ED 13
EX	DE,IY	L	ED 1B
EX	E,E'		CB 33
EX	H,H'		CB 34
EX	HL,HL'	L	ED CB 33
EX	HL,IX	L	ED 33
EX	HL,IY	L	ED 3B
EX	IX,IX'		ED CB 34
EX	IX,IY	L	ED 2B
EX	IY,IY'	L	ED CB 35
EX	L,L'		CB 35
EXALL			ED D9
EXTS	A	L	ED 65
EXTS		L	ED 65
EXTSW	HL		ED 75
EXTSW			ED 75
EXX			D9
EXXX			DD D9
EXXY			FD D9
HALT			76
IM	0		ED 46
IM	1		ED 56
IM	2		ED 5E
IM	3		ED 4E
IN	A,(12H)		DB 12
IN	A,(C)		ED 78
IN	B,(C)		ED 40
IN	C,(C)		ED 48
IN	D,(C)		ED 50
IN	E,(C)		ED 58
IN	H,(C)		ED 60
IN	L,(C)		ED 68
IN0	(12H)		ED 30 12
IN0	A,(12H)		ED 38 12
IN0	B,(12H)		ED 00 12
IN0	C,(12H)		ED 08 12
IN0	D,(12H)		ED 10 12
IN0	E,(12H)		ED 18 12
IN0	H,(12H)		ED 20 12
IN0	L,(12H)		ED 28 12
INA	A,(1234H)	I	ED DB 34 12
INAW	HL,(1234H)	I	FD DB 34 12
INC	(HL)		34
INC	(IX+12H)	I	DD 34 12
INC	(IY+12H)	I	FD 34 12
INC	A		3C

Source Code	Mode	Object Code
INC B		04
INC BC	X	03
INC C		0C
INC D		14
INC DE	X	13
INC E		1C
INC H		24
INC HL	X	23
INC IX	X	DD 23
INC IXL		DD 2C
INC IXU		DD 24
INC IY	X	FD 23
INC IYL		FD 2C
INC IYU		FD 24
INC	L	2C
INC SP	X	33
INCW BC	X	03
INCW DE	X	13
INCW HL	X	23
INCW IX	X	DD 23
INCW IY	X	FD 23
INCW SP	X	33
IND		ED AA
INDR		ED BA
INDRW		ED FA
INDW		ED EA
INI		ED A2
INIR		ED B2
INIRW		ED F2
INIW		ED E2
INW BC,(C)		DD 40
INW DE,(C)		DD 50
INW HL,(C)		DD 78
JP (HL)	X	E9
JP (IX)	X	DD E9
JP (IY)	X	FD E9
JP 1234H	I X	C3 34 12
JP C,1234H	I X	DA 34 12
JP M,1234H	I X	FA 34 12
JP NC,1234H	I X	D2 34 12
JP NZ,1234H	I X	C2 34 12
JP NS,1234H	I X	F2 34 12
JP NV,1234H	I X	E2 34 12
JP P,1234H	I X	F2 34 12
JP PE,1234H	I X	EA 34 12
JP PO,1234H	I X	E2 34 12
JP S,1234H	I X	FA 34 12
JP V,1234H	I X	E2 34 12
JP Z,1234H	I X	CA 34 12
JR 123456H	X	FD 18 56 34 12
JR 1234H	X	DD 18 34 12
JR 12H	X	18 12
JR C,123456H	X	FD 38 56 34 12
JR C,1234H	X	DD 38 34 12
JR C,12H	X	38 12
JR NC,123456H	X	FD 30 56 34 12
JR NC,1234H	X	DD 30 34 12
JR NC,12H	X	30 12
JR NZ,123456H	X	FD 20 56 34 12
JR NZ,1234H	X	DD 20 34 12
JR NZ,12H	X	20 12
JR NZ,12H	X	20 12
JR Z,123456H	X	FD 28 56 34 12
JR Z,1234H	X	DD 28 34 12
JR Z,12H	X	28 12
LD (1234H),A	I	32 34 12
LD (1234H),BC	I L	ED 43 34 12
LD (1234H),DE	I L	ED 53 34 12
LD (1234H),HL	I L	22 34 12
LD (1234H),HL	I L	ED 63 34 12
LD (1234H),IX	I L	DD 22 34 12
LD (1234H),IY	I L	FD 22 34 12
LD (1234H),SP	I L	ED 73 34 12
LD (BC),A		02
LD (BC),BC	L	FD 0C
LD (BC),DE	L	FD 1C
LD (BC),HL	L	FD 3C
LD (BC),IX	L	DD 01
LD (BC),IY	L	FD 01
LD (DE),A		12
LD (DE),BC	L	FD 0D
LD (DE),DE	L	FD 1D
LD (DE),HL	L	FD 3D
LD (DE),IX	L	DD 11
LD (DE),IY	L	FD 11
LD (HL),12H		36 12
LD (HL),A		77
LD (HL),B		70
LD (HL),BC	L	FD 0F
LD (HL),C		71
LD (HL),D		72
LD (HL),DE	L	FD 1F
LD (HL),E		73
LD (HL),H		74
LD (HL),HL	L	FD 3F
LD (HL),IX	L	DD 31
LD (HL),IY	L	FD 31
LD (HL),L		75
LD (IX+12H),34H	I	DD 36 12 34
LD (IX+12H),A	I	DD 77 12
LD (IX+12H),B	I	DD 70 12
LD (IX+12H),BC	I L	DD CB 12 0B
LD (IX+12H),C	I	DD 71 12
LD (IX+12H),D	I	DD 72 12
LD (IX+12H),E	I	DD 73 12
LD (IX+12H),DE	I L	DD CB 12 1B
LD (IX+12H),H	I	DD 74 12
LD (IX+12H),HL	I L	DD CB 12 3B

Source Code		Mode		Object Code			
LD	(IX+12H),IY	I	L	DD	CB	12	2B
LD	(IX+12H),L	I		DD	75	12	
LD	(IY+12H),34H	I		FD	36	34	12
LD	(IY+12H),A	I		FD	77	12	
LD	(IY+12H),B	I		FD	70	12	
LD	(IY+12H),BC	I	L	FD	CB	12	0B
LD	(IY+12H),C	I		FD	71	12	
LD	(IY+12H),D	I		FD	72	12	
LD	(IY+12H),DE	I		FD	CB	12	1B
LD	(IY+12H),E	I	L	FD	73	12	
LD	(IY+12H),H	I		FD	74	12	
LD	(IY+12H),HL	I	L	FD	CB	12	3B
LD	(IY+12H),IX	I	L	FD	CB	12	2B
LD	(IY+12H),L	I		FD	75	12	
LD	(SP+12H),BC	I	L	DD	CB	12	09
LD	(SP+12H),DE	I	L	DD	CB	12	19
LD	(SP+12H),HL	I	L	DD	CB	12	39
LD	(SP+12H),IX	I	L	DD	CB	12	29
LD	(SP+12H),IY	I	L	FD	CB	12	29
LD	A,(1234H)	I		3A	34	12	
LD	A,(BC)			0A			
LD	A,(DE)			1A			
LD	A,(HL)			7E			
LD	A,(IX+12H)	I		DD	7E	12	
LD	A,(IY+12H)	I		FD	7E	12	
LD	A,12H			3E	12		
LD	A,A			7F			
LD	A,B			78			
LD	A,C			79			
LD	A,D			7A			
LD	A,E			7B			
LD	A,H			7C			
LD	A,I			ED	57		
LD	A,IXL			DD	7D		
LD	A,IXU			DD	7C		
LD	A,IYL			FD	7D		
LD	A,IYU			FD	7C		
LD	A,L			7D			
LD	A,R			ED	5F		
LD	B,(HL)			46			
LD	B,(IX+12H)	I		DD	46	12	
LD	B,(IY+12H)	I		FD	46	12	
LD	B,12H			06	12		
LD	B,A			47			
LD	B,B			40			
LD	B,C			41			
LD	B,D			42			
LD	B,E			43			
LD	B,H			44			
LD	B,IXL			DD	45		
LD	B,IXU			DD	44		
LD	B,IYL			FD	45		
LD	B,IYU			FD	44		
LD	B,L			45			
LD	BC,(1234H)	I	L	ED	4B	34	12
LD	BC,(BC)		L	DD	0C		
LD	BC,(DE)		L	DD	0D		
LD	BC,(HL)		L	DD	0F		
LD	BC,(IX+12H)	I	L	DD	CB	12	03
LD	BC,(IY+12H)	I	L	FD	CB	12	03
LD	BC,(SP+12H)	I	L	DD	CB	12	01
LD	BC,1234H	I	L	01	34	12	
LD	BC,BC		L	ED	02		
LD	BC,DE		L	DD	02		
LD	BC,HL		L	FD	02		
LD	BC,IX		L	DD	0B		
LD	BC,IY		L	FD	0B		
LD	C,(HL)			4E			
LD	C,(IX+12H)	I		DD	4E	12	
LD	C,(IY+12H)	I		FD	4E	12	
LD	C,12H			0E	12		
LD	C,A			4F			
LD	C,B			48			
LD	C,C			49			
LD	C,D			4A			
LD	C,E			4B			
LD	C,H			4C			
LD	C,IXL			DD	4D		
LD	C,IXU			DD	4C		
LD	C,IYL			FD	4D		
LD	C,IYU			FD	4C		
LD	C,L			4D			
LD	D,(HL)			56			
LD	D,(IX+12H)	I		DD	56	12	
LD	D,(IY+12H)	I		FD	56	12	
LD	D,12H			16	12		
LD	D,A			57			
LD	D,B			50			
LD	D,C			51			
LD	D,D			52			
LD	D,E			53			
LD	D,H			54			
LD	D,IXL			DD	55		
LD	D,IXU			DD	54		
LD	D,IYL			FD	55		
LD	D,IYU			FD	54		
LD	D,L			55			
LD	DE,(1234H)	I	L	ED	5B	34	12
LD	DE,(BC)		L	DD	1C		
LD	DE,(DE)		L	DD	1D		
LD	DE,(HL)		L	DD	1F		
LD	DE,(IX+12H)	I	L	DD	CB	12	13
LD	DE,(IY+12H)	I	L	FD	CB	12	13
LD	DE,(SP+12H)	I	L	DD	CB	12	11
LD	DE,1234H	I	L	11	34	12	
LD	DE,BC		L	ED	12		
LD	DE,DE		L	DD	12		
LD	DE,HL		L	FD	12		

Source Code	Mode		Object Code
LD DE,IX		L	DD 1B
LD DE,IY		L	FD 1B
LD E,(HL)			5E
LD E,(IX+12H)	I		DD 5E 12
LD E,(IY+12H)	I		FD 5E 12
LD E,12H			1E 12
LD E,A			5F
LD E,B			58
LD E,C			59
LD E,D			5A
LD E,E			5B
LD E,H			5C
LD E,L			5D
LD E,IXL			DD 5D
LD E,IYU			FD 5C
LD E,IYL			DD 5D
LD E,IYU			FD 5D
LD H,(HL)			66
LD H,(IX+12H)	I		DD 66 12
LD H,(IY+12H)	I		FD 66 12
LD H,12H			26 12
LD H,A			67
LD H,B			60
LD H,C			61
LD H,D			62
LD H,E			63
LD H,H			64
LD H,L			65
LD HL,(1234H)	I	L	2A 34 12
LD HL,(1234H)	I	L	ED 6B 34 12
LD HL,(BC)		L	DD 3C
LD HL,(DE)		L	DD 3D
LD HL,(HL)		L	DD 3F
LD HL,(IX+12H)	I	L	DD CB 12 33
LD HL,(IY+12H)	I	L	FD CB 12 33
LD HL,(SP+12H)	I	L	DD CB 12 31
LD HL,1234H	I	L	21 34 12
LD HL,BC		L	ED 32
LD HL,DE		L	DD 32
LD HL,HL		L	FD 32
LD HL,I		L	DD 57
LD HL,IX		L	DD 3B
LD HL,IY		L	FD 3B
LD I,A			ED 47
LD I,HL		L	DD 47
LD IX,(1234H)	I	L	DD 2A 34 12
LD IX,(BC)		L	DD 03
LD IX,(DE)		L	DD 13
LD IX,(HL)		L	DD 33
LD IX,(IY+12H)	I	L	FD CB 12 23
LD IX,(SP+12H)	I	L	DD CB 12 21
LD IX,1234H	I	L	DD 21 34 12
LD IX,BC		L	DD 07
LD IX,DE		L	DD 17

Source Code	Mode		Object Code
LD IX,HL		L	DD 37
LD IX,IY		L	DD 27
LD IXL,12H			DD 2E 12
LD IXL,A			DD 6F
LD IXL,B			DD 68
LD IXL,C			DD 69
LD IXL,D			DD 6A
LD IXL,E			DD 6B
LD IXL,IXL			DD 6D
LD IXL,IXU			DD 6C
LD IXU,12H			DD 26 12
LD IXU,A			DD 67
LD IXU,B			DD 60
LD IXU,C			DD 61
LD IXU,D			DD 62
LD IXU,E			DD 63
LD IXU,IXL			DD 65
LD IXU,IXU			DD 64
LD IY,(1234H)	I	L	FD 2A 34 12
LD IY,(BC)		L	FD 03
LD IY,(DE)		L	FD 13
LD IY,(HL)		L	FD 33
LD IY,(IX+12H)	I	L	DD CB 12 23
LD IY,(SP+12H)	I	L	FD CB 12 21
LD IY,1234H	I	L	FD 21 34 12
LD IY,BC		L	FD 07
LD IY,DE		L	FD 17
LD IY,HL		L	FD 37
LD IY,IX		L	FD 27
LD IYL,12H			FD 2E 12
LD IYL,A			FD 6F
LD IYL,B			FD 68
LD IYL,C			FD 69
LD IYL,D			FD 6A
LD IYL,E			FD 6B
LD IYL,IYL			FD 6D
LD IYL,IYU			FD 6C
LD IYU,12H			FD 26 12
LD IYU,A			FD 67
LD IYU,B			FD 60
LD IYU,C			FD 61
LD IYU,D			FD 62
LD IYU,E			FD 63
LD IYU,IYL			FD 65
LD IYU,IYU			FD 64
LD L,(HL)			6E
LD L,(IX+12H)	I		DD 6E 12
LD L,(IY+12H)	I		FD 6E 12
LD L,12H			2E 12
LD L,A			6F
LD L,B			68
LD L,C			69
LD L,D			6A
LD L,E			6B

Source Code		Mode		Object Code				
LD	L,H			6C				
LD	L,L			6D				
LD	R,A			ED	4F			
LD	SP,(1234H)	I	L	ED	7B	34	12	
LD	SP,1234H	I	L	31	34	12		
LD	SP,HL		L	F9				
LD	SP,IX		L	DD	F9			
LD	SP,IY		L	FD	F9			
LDCTL	A,DSR			ED	D0			
LDCTL	A,XSR			DD	D0			
LDCTL	A,YSR			FD	D0			
LDCTL	DSR,01H			ED	DA	01		
LDCTL	DSR,A			ED	D8			
LDCTL	HL,SR		L	ED	C0			
LDCTL	SR,01H			DD	CA	01		
LDCTL	SR,A			DD	C8			
LDCTL	SR,HL		L	ED	C8			
LDCTL	XSR,01H			DD	DA	01		
LDCTL	XSR,A			DD	D8			
LDCTL	YSR,01H			FD	DA	01		
LDCTL	YSR,A			FD	D8			
LDD				ED	A8			
LDDR				ED	B8			
LDDRW			L	ED	F8			
LDDW			L	ED	E8			
LDI				ED	A0			
LDIR				ED	B0			
LDIRW			L	ED	F0			
LDIW			L	ED	E0			
LDW	(BC),1234H	I	L	ED	06	34	12	
LDW	(DE),1234H	I	L	ED	16	34	12	
LDW	(HL),1234H	I	L	ED	36	34	12	
LDW	HL,I		L	DD	57			
LDW	I,HL		L	DD	47			
MLT	BC			ED	4C			
MLT	DE			ED	5C			
MLT	HL			ED	6C			
MLT	SP			ED	7C			
MTEST				DD	CF			
MULTUW	(IX+12H)	I		DD	CB	12	9A	
MULTUW	(IY+12H)	I		FD	CB	12	9A	
MULTUW	1234H			ED	CB	9F		
MULTUW	BC			ED	CB	98		
MULTUW	DE			ED	CB	99		
MULTUW	HL			ED	CB	9B		
MULTUW	HL,(IX+12H)	I		DD	CB	12	9A	
MULTUW	HL,(IY+12H)	I		FD	CB	12	9A	
MULTUW	HL,1234H			ED	CB	9F		
MULTUW	HL,BC			ED	CB	98		
MULTUW	HL,DE			ED	CB	99		
MULTUW	HL,HL			ED	CB	9B		
MULTUW	HL,IX			ED	CB	9C		
MULTUW	HL,IY			ED	CB	9D		
MULTUW	IX			ED	CB	9C		
MULTUW	IY			ED	CB	9D		
MULTW	(IX+12H)	I		DD	CB	12	92	
MULTW	(IY+12H)	I		FD	CB	12	92	
MULTW	1234H			ED	CB	97	34	12
MULTW	BC			ED	CB	90		
MULTW	DE			ED	CB	91		
MULTW	HL			ED	CB	93		
MULTW	HL,(IX+12H)	I		DD	CB	12	92	
MULTW	HL,(IY+12H)	I		FD	CB	12	92	
MULTW	HL,1234H			ED	CB	97	34	12
MULTW	HL,BC			ED	CB	90		
MULTW	HL,DE			ED	CB	91		
MULTW	HL,HL			ED	CB	93		
MULTW	HL,IX			ED	CB	94		
MULTW	HL,IY			ED	CB	95		
MULTW	IX			ED	CB	94		
MULTW	IY			ED	CB	95		
NEG	A			ED	44			
NEG				ED	44			
NEGW	HL			ED	54			
NEGW				ED	54			
NOP				00				
OR	(HL)			B6				
OR	(IX+12H)	I		DD	B6	12		
OR	(IY+12H)	I		FD	B6	12		
OR	12H			F6	12			
OR	A			B7				
OR	A,(HL)			B6				
OR	A,(IX+12H)	I		DD	B6	12		
OR	A,(IY+12H)	I		FD	B6	12		
OR	A,12H			F6	12			
OR	A,A			B7				
OR	A,B			B0				
OR	A,C			B1				
OR	A,D			B2				
OR	A,E			B3				
OR	A,H			B4				
OR	A,IXL			DD	B5			
OR	A,IXU			DD	B4			
OR	A,IYL			FD	B5			
OR	A,IYU			FD	B4			
OR	A,L			B5				
OR	B			B0				
OR	C			B1				
OR	D			B2				
OR	E			B3				
OR	H			B4				
OR	IXL			DD	B5			
OR	IXU			DD	B4			
OR	IYL			FD	B5			
OR	IYU			FD	B4			
OR	L			B5				
ORW	(IX+12H)	I		DD	F6	12		
ORW	(IY+12H)	I		FD	F6	12		
ORW	1234H			ED	B6	34	12	
ORW	BC			ED	B4			

Source Code		Mode		Object Code			
ORW	DE			ED	B5		
ORW	HL			ED	B7		
ORW	HL,(IX+12H)	I		DD	F6	12	
ORW	HL,(IY+12H)	I		FD	F6	12	
ORW	HL,1234H			ED	B6	34	12
ORW	HL,BC			ED	B4		
ORW	HL,DE			ED	B5		
ORW	HL,HL			ED	B7		
ORW	HL,IX			DD	B7		
ORW	HL,IY			FD	B7		
ORW	IX			DD	B7		
ORW	IY			FD	B7		
OTDM				ED	8B		
OTDMR				ED	9B		
OTDR				ED	BB		
OTDRW				ED	FB		
OTIM				ED	83		
OTIMR				ED	93		
OTIR				ED	B3		
OTIRW				ED	F3		
OUT	(12H),A			D3	12		
OUT	(C),12H			ED	71	12	
OUT	(C),A			ED	79		
OUT	(C),B			ED	41		
OUT	(C),C			ED	49		
OUT	(C),D			ED	51		
OUT	(C),E			ED	59		
OUT	(C),H			ED	61		
OUT	(C),L			ED	69		
OUT0	(12H),A			ED	39	12	
OUT0	(12H),B			ED	01	12	
OUT0	(12H),C			ED	09	12	
OUT0	(12H),D			ED	11	12	
OUT0	(12H),E			ED	19	12	
OUT0	(12H),H			ED	21	12	
OUT0	(12H),L			ED	29	12	
OUTA	(1234H),A	I		ED	D3	34	12
OUTAW	(1234H),HL	I		FD	D3	34	12
OUTD				ED	AB		
OUTDW				ED	EB		
OUTI				ED	A3		
OUTIW				ED	E3		
OUTW	(C),1234H			FD	79	34	12
OUTW	(C),BC			DD	41		
OUTW	(C),DE			DD	51		
OUTW	(C),HL			DD	79		
POP	AF		L	F1			
POP	BC		L	C1			
POP	DE		L	D1			
POP	HL		L	E1			
POP	IX		L	DD	E1		
POP	IY		L	FD	E1		
POP	SR		L	ED	C1		
PUSH	1234H	I	L	FD	F5	34	12

Source Code		Mode		Object Code			
PUSH	AF		L	F5			
PUSH	BC		L	C5			
PUSH	DE		L	D5			
PUSH	HL		L	E5			
PUSH	IX		L	DD	E5		
PUSH	IY		L	FD	E5		
PUSH	SR		L	ED	C5		
RES	0,(HL)			CB	86		
RES	0,(IX+12H)	I		DD	CB	12	86
RES	0,(IY+12H)	I		FD	CB	12	86
RES	0,A			CB	87		
RES	0,B			CB	80		
RES	0,C			CB	81		
RES	0,D			CB	82		
RES	0,E			CB	83		
RES	0,H			CB	84		
RES	0,L			CB	85		
RES	1,(HL)			CB	8E		
RES	1,(IX+12H)	I		DD	CB	12	8E
RES	1,(IY+12H)	I		FD	CB	12	8E
RES	1,A			CB	8F		
RES	1,B			CB	88		
RES	1,C			CB	89		
RES	1,D			CB	8A		
RES	1,E			CB	8B		
RES	1,H			CB	8C		
RES	1,L			CB	8D		
RES	2,(HL)			CB	96		
RES	2,(IX+12H)	I		DD	CB	12	96
RES	2,(IY+12H)	I		FD	CB	12	96
RES	2,A			CB	97		
RES	2,B			CB	90		
RES	2,C			CB	91		
RES	2,D			CB	92		
RES	2,E			CB	93		
RES	2,H			CB	94		
RES	2,L			CB	95		
RES	3,(HL)			CB	9E		
RES	3,(IX+12H)	I		DD	CB	12	9E
RES	3,(IY+12H)	I		FD	CB	12	9E
RES	3,A			CB	9F		
RES	3,B			CB	98		
RES	3,C			CB	99		
RES	3,D			CB	9A		
RES	3,E			CB	9B		
RES	3,H			CB	9C		
RES	3,L			CB	9D		
RES	4,(HL)			CB	A6		
RES	4,(IX+12H)	I		DD	CB	12	A6
RES	4,(IY+12H)	I		FD	CB	12	A6
RES	4,A			CB	A7		
RES	4,B			CB	A0		
RES	4,C			CB	A1		
RES	4,D			CB	A2		

Source Code	Mode	Object Code
RES 4,E		CB A3
RES 4,H		CB A4
RES 4,L		CB A5
RES 5,(HL)		CB AE
RES 5,(IX+12H)	I	DD CB 12 AE
RES 5,(IY+12H)	I	FD CB 12 AE
RES 5,A		CB AF
RES 5,B		CB A8
RES 5,C		CB A9
RES 5,D		CB AA
RES 5,E		CB AB
RES 5,H		CB AC
RES 5,L		CB AD
RES 6,(HL)		CB B6
RES 6,(IX+12H)	I	DD CB 12 B6
RES 6,(IY+12H)	I	FD CB 12 B6
RES 6,A		CB B7
RES 6,B		CB B0
RES 6,C		CB B1
RES 6,D		CB B2
RES 6,E		CB B3
RES 6,H		CB B4
RES 6,L		CB B5
RES 7,(HL)		CB BE
RES 7,(IX+12H)	I	DD CB 12 BE
RES 7,(IY+12H)	I	FD CB 12 BE
RES 7,A		CB BF
RES 7,B		CB B8
RES 7,C		CB B9
RES 7,D		CB BA
RES 7,E		CB BB
RES 7,H		CB BC
RES 7,L		CB BD
RESC LCK		ED FF
RESC LW		DD FF
reserved		ED 55
RET C	X	D8
RET M	X	F8
RET NC	X	D0
RET NS	X	F0
RET NV	X	E0
RET NZ	X	C0
RET P	X	F0
RET PE	X	E8
RET PO	X	E0
RET S	X	F8
RET V	X	E8
RET Z	X	C8
RET	X	C9
RETI	X	ED 4D
RETN	X	ED 45
RL (HL)		CB 16
RL (IX+12H)	I	DD CB 12 16
RL (IY+12H)	I	FD CB 12 16
RL A		CB 17
RL B		CB 10
RL C		CB 11
RL D		CB 12
RL E		CB 13
RL H		CB 14
RL L		CB 15
RLA		17
RLC (HL)		CB 06
RLC (IX+12H)	I	DD CB 12 06
RLC (IY+12H)	I	FD CB 12 06
RLC A		CB 07
RLC B		CB 00
RLC C		CB 01
RLC D		CB 02
RLC E		CB 03
RLC H		CB 04
RLC L		CB 05
RLCA		07
RLCW (HL)		ED CB 02
RLCW (IX+12H)	I	DD CB 12 02
RLCW (IY+12H)	I	FD CB 12 02
RLCW BC		ED CB 00
RLCW DE		ED CB 01
RLCW HL		ED CB 03
RLCW IX		ED CB 04
RLCW IY		ED CB 05
RLD		ED 6F
RLW (HL)		ED CB 12
RLW (IX+12H)	I	DD CB 12 12
RLW (IY+12H)	I	FD CB 12 12
RLW BC		ED CB 10
RLW DE		ED CB 11
RLW HL		ED CB 13
RLW IX		ED CB 14
RLW IY		ED CB 15
RR (HL)		CB 1E
RR (IX+12H)	I	DD CB 12 1E
RR (IY+12H)	I	FD CB 12 1E
RR A		CB 1F
RR B		CB 18
RR C		CB 19
RR D		CB 1A
RR E		CB 1B
RR H		CB 1C
RR L		CB 1D
RRA		1F
RRC (HL)		CB 0E
RRC (IX+12H)	I	DD CB 12 0E
RRC (IY+12H)	I	FD CB 12 0E
RRC A		CB 0F
RRC B		CB 08
RRC C		CB 09
RRC D		CB 0A
RRC E		CB 0B

Source Code	Mode	Object Code
RRC H		CB 0C
RRC L		CB 0D
RRCA		0F
RRCW (HL)		ED CB 0A
RRCW (IX+12H)	I	DD CB 12 0A
RRCW (IY+12H)	I	FD CB 12 0A
RRCW BC		ED CB 08
RRCW DE		ED CB 09
RRCW HL		ED CB 0B
RRCW IX		ED CB 0C
RRCW IY		ED CB 0D
RRD		ED 67
RRW (HL)		ED CB 1A
RRW (IX+12H)	I	DD CB 12 1A
RRW (IY+12H)	I	FD CB 12 1A
RRW BC		ED CB 18
RRW DE		ED CB 19
RRW HL		ED CB 1B
RRW IX		ED CB 1C
RRW IY		ED CB 1D
RST 00H	X	C7
RST 08H	X	CF
RST 10H	X	D7
RST 18H	X	DF
RST 20H	X	E7
RST 28H	X	EF
RST 30H	X	F7
RST 38H	X	FF
SBC A,(HL)		9E
SBC A,(IX+12H)	I	DD 9E 12
SBC A,(IY+12H)	I	FD 9E 12
SBC A,12H		DE 12
SBC A,A		9F
SBC A,B		98
SBC A,C		99
SBC A,D		9A
SBC A,E		9B
SBC A,H		9C
SBC A,IXL		DD 9D
SBC A,IXU		DD 9C
SBC A,IYL		FD 9D
SBC A,IYU		FD 9C
SBC A,L		9D
SBC HL,BC		ED 42
SBC HL,DE		ED 52
SBC HL,HL		ED 62
SBC HL,SP		ED 72
SBCW (IX+12H)	I	DD DE 12
SBCW (IY+12H)	I	FD DE 12
SBCW 1234H		ED 9E 34 12
SBCW BC		ED 9C
SBCW DE		ED 9D
SBCW HL		ED 9F
SBCW HL,(IX+12H)		DD DE 12
SBCW HL,(IY+12H)		FD DE 12
SBCW HL,1234H		ED 9E 34 12
SBCW HL,BC		ED 9C
SBCW HL,DE		ED 9D
SBCW HL,HL		ED 9F
SBCW HL,IX		DD 9F
SBCW HL,IY		FD 9F
SBCW IX		DD 9F
SBCW IY		FD 9F
SCF		37
SET 0,(HL)		CB C6
SET 0,(IX+12H)	I	DD CB 12 C6
SET 0,(IY+12H)	I	FD CB 12 C6
SET 0,A		CB C7
SET 0,B		CB C0
SET 0,C		CB C1
SET 0,D		CB C2
SET 0,E		CB C3
SET 0,H		CB C4
SET 0,L		CB C5
SET 1,(HL)		CB CE
SET 1,(IX+12H)	I	DD CB 12 CE
SET 1,(IY+12H)	I	FD CB 12 CE
SET 1,A		CB CF
SET 1,B		CB C8
SET 1,C		CB C9
SET 1,D		CB CA
SET 1,E		CB CB
SET 1,H		CB CC
SET 1,L		CB CD
SET 2,(HL)		CB D6
SET 2,(IX+12H)	I	DD CB 12 D6
SET 2,(IY+12H)	I	FD CB 12 D6
SET 2,A		CB D7
SET 2,B		CB D0
SET 2,C		CB D1
SET 2,D		CB D2
SET 2,E		CB D3
SET 2,H		CB D4
SET 2,L		CB D5
SET 3,(HL)		CB DE
SET 3,(IX+12H)	I	DD CB 12 DE
SET 3,(IY+12H)	I	FD CB 12 DE
SET 3,A		CB DF
SET 3,B		CB D8
SET 3,C		CB D9
SET 3,D		CB DA
SET 3,E		CB DB
SET 3,H		CB DC
SET 3,L		CB DD
SET 4,(HL)		CB E6
SET 4,(IX+12H)	I	DD CB 12 E6
SET 4,(IY+12H)	I	FD CB 12 E6
SET 4,A		CB E7

Source Code	Mode	Object Code
SET 4,B		CB E0
SET 4,C		CB E1
SET 4,D		CB E2
SET 4,E		CB E3
SET 4,H		CB E4
SET 4,L		CB E5
SET 5,(HL)		CB EE
SET 5,(IX+12H)	I	DD CB 12 EE
SET 5,(IY+12H)	I	FD CB 12 EE
SET 5,A		CB EF
SET 5,B		CB E8
SET 5,C		CB E9
SET 5,D		CB EA
SET 5,E		CB EB
SET 5,H		CB EC
SET 5,L		CB ED
SET 6,(HL)		CB F6
SET 6,(IX+12H)	I	DD CB 12 F6
SET 6,(IY+12H)	I	FD CB 12 F6
SET 6,A		CB F7
SET 6,B		CB F0
SET 6,C		CB F1
SET 6,D		CB F2
SET 6,E		CB F3
SET 6,H		CB F4
SET 6,L		CB F5
SET 7,(HL)		CB FE
SET 7,(IX+12H)	I	DD CB 12 FE
SET 7,(IY+12H)	I	FD CB 12 FE
SET 7,A		CB FF
SET 7,B		CB F8
SET 7,C		CB F9
SET 7,D		CB FA
SET 7,E		CB FB
SET 7,H		CB FC
SET 7,L		CB FD
SETC LCK		ED F7
SETC LW		DD F7
SETC XM		FD F7
SLA (HL)		CB 26
SLA (IX+12H)	I	DD CB 12 26
SLA (IY+12H)	I	FD CB 12 26
SLA A		CB 27
SLA B		CB 20
SLA C		CB 21
SLA D		CB 22
SLA E		CB 23
SLA H		CB 24
SLA L		CB 25
SLAW (HL)		ED CB 22
SLAW (IX+12H)	I	DD CB 12 22
SLAW (IY+12H)	I	FD CB 12 22
SLAW BC		ED CB 20
SLAW DE		ED CB 21

Source Code	Mode	Object Code
SLAW HL		ED CB 23
SLAW IX		ED CB 24
SLAW IY		ED CB 25
SLP		ED 76
SRA (HL)		CB 2E
SRA (IX+12H)	I	DD CB 12 2E
SRA (IY+12H)	I	FD CB 12 2E
SRA A		CB 2F
SRA B		CB 28
SRA C		CB 29
SRA D		CB 2A
SRA E		CB 2B
SRA H		CB 2C
SRA L		CB 2D
SRAW (HL)		ED CB 2A
SRAW (IX+12H)	I	DD CB 12 2A
SRAW (IY+12H)	I	FD CB 12 2A
SRAW BC		ED CB 28
SRAW DE		ED CB 29
SRAW HL		ED CB 2B
SRAW IX		ED CB 2C
SRAW IY		ED CB 2D
SRL (HL)		CB 3E
SRL (IX+12H)	I	DD CB 12 3E
SRL (IY+12H)	I	FD CB 12 3E
SRL A		CB 3F
SRL B		CB 38
SRL C		CB 39
SRL D		CB 3A
SRL E		CB 3B
SRL H		CB 3C
SRL L		CB 3D
SRLW (HL)		ED CB 3A
SRLW (IX+12H)	I	DD CB 12 3A
SRLW (IY+12H)	I	FD CB 12 3A
SRLW BC		ED CB 38
SRLW DE		ED CB 39
SRLW HL		ED CB 3B
SRLW IX		ED CB 3C
SRLW IY		ED CB 3D
SUB A,(HL)		96
SUB A,12H		D6 12
SUB A,A		97
SUB A,(IX+12H)	I	DD 96 12
SUB A,(IY+12H)	I	FD 96 12
SUB 12H		D6 12
SUB A,B		90
SUB A,C		91
SUB A,D		92
SUB A,E		93
SUB A,H		94
SUB A,IXL		DD 95
SUB A,IXU		DD 94
SUB A,IYL		FD 95

Source Code	Mode	Object Code
SUB A,IYU		FD 94
SUB A,L		95 SUB
HL,(1234H)	I X	ED D6 34 12
SUB SP,1234H	I X	ED 92 34 12
SUBW (IX+12H)		DD D6 12
SUBW (IY+12H)		FD D6 12
SUBW 1234H		ED 96 34 12
SUBW BC		ED 94
SUBW DE		ED 95
SUBW HL		ED 97
SUBW HL,(IX+12H)	I	DD D6 12
SUBW HL,(IY+12H)	I	FD D6 12
SUBW HL,1234H		ED 96 34 12
SUBW HL,BC		ED 94
SUBW HL,DE		ED 95
SUBW HL,HL		ED 97
SUBW HL,IX		DD 97
SUBW HL,IY		FD 97
SUBW IX		DD 97
SUBW IY		FD 97
SWAP BC		ED 0E
SWAP DE		ED 1E
SWAP HL		ED 3E
SWAP IX		DD 3E
SWAP IY		FD 3E
TST (HL)		ED 34
TST 12H		ED 64 12
TST A		ED 3C
TST B		ED 04
TST C		ED 0C
TST D		ED 14
TST E		ED 1C
TST H		ED 24
TST L		ED 2C
TSTIO 12H		ED 74 12
XOR (HL)		AE
XOR (IX+12H)	I	DD AE 12
XOR (IY+12H)	I	FD AE 12
XOR 12H		EE 12
XOR A		AF
XOR A,(HL)		AE
XOR A,(IX+12H)	I	DD AE 12
XOR A,(IY+12H)	I	FD AE 12
XOR A,12H		EE 12
XOR A,A		AF
XOR A,B		A8
XOR A,C		A9
XOR A,D		AA
XOR A,E		AB
XOR A,H		AC
XOR A,IXL		DD AD
XOR A,IXU		DD AC
XOR A,IYL		FD AD
XOR A,IYU		FD AC
XOR A,L		AD
XOR B		A8
XOR C		A9
XOR D		AA
XOR E		AB
XOR H		AC
XOR IXL		DD AD
XOR IXU		DD AC
XOR IYL		FD AD
XOR IYU		FD AC
XOR L		AD
XORW (IX+12H)	I	DD EE 12
XORW (IY+12H)	I	FD EE 12
XORW 1234H		ED AE 34 12
XORW BC		ED AC
XORW DE		ED AD
XORW HL		ED AF
XORW HL,(IX+12H)	I	DD EE 12
XORW HL,(IY+12H)	I	FD EE 12
XORW HL,1234H		ED AE 34 12
XORW HL,BC		ED AC
XORW HL,DE		ED AD
XORW HL,HL		ED AF
XORW HL,IX		DD AF
XORW HL,IY		FD AF
XORW IX		DD AF
XORW IY		FD AF

Appendix A A

Appendix B B

Appendix C C

Appendix D D

Appendix E E

Index I

Superintegration™ Products Guide S

Zilog's Literature Guide Ordering Information L

APPENDIX C

Z380™ INSTRUCTION IN NUMERIC ORDER

The following Appendix has the Z380 instructions sorted by numeric order.

The column "Mode" indicates whether the instruction is affected by DDIR immediate Decoder Directives, Extended mode or Native mode of operation, and Word or Long Word Mode of operation; "I" means the instruction can be used with DDIR IM to expand its immediate constant, "X" means that the operation of the instruction is affected by the XM status bit, and "L" means that the instruction is affected by LW status bit, or can be used with DDIR LW or DDIR W. The Native/Extended modes, Word/Long Word modes and Decoder Directives are discussed in Chapter 3 in this manual.

Object Code	Source Code		Mode		
00	NOP				
01 34 12	LD	BC,1234H	I		L
02	LD	(BC),A			
03	INC	BC		X	
03	INCW	BC		X	
04	INC	B			
05	DEC	B			
06 12	LD	B,12H			
07	RLCA				
08	EX	AF,AF'			
09	ADD	HL,BC		X	
0A	LD	A,(BC)			
0B	DEC	BC		X	
0B	DECW	BC		X	
0C	INC	C			
0D	DEC	C			
0E 12	LD	C,12H			
0F	RRCA				
10 12	DJNZ	12H		X	
11 34 12	LD	DE,1234H	I		L
12	LD	(DE),A			
13	INC	DE		X	
13	INCW	DE		X	
14	INC	D			
15	DEC	D			
16 12	LD	D,12H			
17	RLA				
18 12	JR	12H		X	
19	ADD	HL,DE		X	
1A	LD	A,(DE)			
1B	DEC	DE		X	
1B	DECW	DE		X	
1C	INC	E			
1D	DEC	E			
1E 12	LD	E,12H			
1F	RRA				
20 12	JR	NZ,12H		X	
21 34 12	LD	HL,1234H	I		L
22 34 12	LD	(1234H),HL	I		L
23	INC	HL		X	
23	INCW	HL		X	
24	INC	H			
25	DEC	H			
26 12	LD	H,12H			
27	DAA				
28 12	JR	Z,12H		X	
29	ADD	HL,HL		X	
2A 34 12	LD	HL,(1234H)	I		L
2B	DEC	HL		X	
2B	DECW	HL		X	
2C	INC	L			
2D	DEC	L			
2E 12	LD	L,12H			
2F	CPL	A			
2F	CPL				
30 12	JR	NC,12H		X	
31 34 12	LD	SP,1234H	I		L
32 34 12	LD	(1234H),A	I		
33	INC	SP		X	
33	INCW	SP		X	
34	INC	(HL)			
35	DEC	(HL)			
36 12	LD	(HL),12H			
37	SCF				
38 12	JR	C,12H		X	
39	ADD	HL,SP		X	
3A 34 12	LD	A,(1234H)	I		
3B	DEC	SP		X	
3B	DECW	SP		X	
3C	INC	A			
3D	DEC	A			
3E 12	LD	A,12H			
3F	CCF				
40	LD	B,B			
41	LD	B,C			
42	LD	B,D			
43	LD	B,E			
44	LD	B,H			
45	LD	B,L			
46	LD	B,(HL)			
47	LD	B,A			
48	LD	C,B			
49	LD	C,C			
4A	LD	C,D			
4B	LD	C,E			
4C	LD	C,H			
4D	LD	C,L			
4E	LD	C,(HL)			
4F	LD	C,A			
50	LD	D,B			
51	LD	D,C			
52	LD	D,D			
53	LD	D,E			
54	LD	D,H			
55	LD	D,L			
56	LD	D,(HL)			
57	LD	D,A			
58	LD	E,B			
59	LD	E,C			
5A	LD	E,D			
5B	LD	E,E			
5C	LD	E,H			
5D	LD	E,L			
5E	LD	E,(HL)			
5F	LD	E,A			
60	LD	H,B			
61	LD	H,C			
62	LD	H,D			

Object Code	Source Code	Mode
63	LD H,E	
64	LD H,H	
65	LD H,L	
66	LD H,(HL)	
67	LD H,A	
68	LD L,B	
69	LD L,C	
6A	LD L,D	
6B	LD L,E	
6C	LD L,H	
6D	LD L,L	
6E	LD L,(HL)	
6F	LD L,A	
70	LD (HL),B	
71	LD (HL),C	
72	LD (HL),D	
73	LD (HL),E	
74	LD (HL),H	
75	LD (HL),L	
76	HALT	
77	LD (HL),A	
78	LD A,B	
79	LD A,C	
7A	LD A,D	
7B	LD A,E	
7C	LD A,H	
7D	LD A,L	
7E	LD A,(HL)	
7F	LD A,A	
80	ADD A,B	
81	ADD A,C	
82	ADD A,D	
83	ADD A,E	
84	ADD A,H	
85	ADD A,L	
86	ADD A,(HL)	
87	ADD A,A	
88	ADC A,B	
89	ADC A,C	
8A	ADC A,D	
8B	ADC A,E	
8C	ADC A,H	
8D	ADC A,L	
8E	ADC A,(HL)	
8F	ADC A,A	
90	SUB A,B	
91	SUB A,C	
92	SUB A,D	
93	SUB A,E	
94	SUB A,H	
95	SUB A,L	
96	SUB A,(HL)	
97	SUB A,A	
98	SBC A,B	

Object Code	Source Code	Mode
99	SBC A,C	
9A	SBC A,D	
9B	SBC A,E	
9C	SBC A,H	
9D	SBC A,L	
9E	SBC A,(HL)	
9F	SBC A,A	
A0	AND A,B	
A0	AND B	
A1	AND A,C	
A1	AND C	
A2	AND A,D	
A2	AND D	
A3	AND A,E	
A3	AND E	
A4	AND A,H	
A4	AND H	
A5	AND A,L	
A5	AND L	
A6	AND (HL)	
A6	AND A,(HL)	
A7	AND A	
A7	AND A,A	
A8	XOR A,B	
A8	XOR B	
A9	XOR A,C	
A9	XOR C	
AA	XOR A,D	
AA	XOR D	
AB	XOR A,E	
AB	XOR E	
AC	XOR A,H	
AC	XOR H	
AD	XOR A,L	
AD	XOR L	
AE	XOR (HL)	
AE	XOR A,(HL)	
AF	XOR A	
AF	XOR A,A	
B0	OR A,B	
B0	OR B	
B1	OR A,C	
B1	OR C	
B2	OR A,D	
B2	OR D	
B3	OR A,E	
B3	OR E	
B4	OR A,H	
B4	OR H	
B5	OR A,L	
B5	OR L	
B6	OR (HL)	
B6	OR A,(HL)	
B7	OR A	

Object Code	Source Code		Mode
B7	OR	A,A	
B8	CP	A,B	
B8	CP	B	
B9	CP	A,C	
B9	CP	C	
BA	CP	A,D	
BA	CP	D	
BB	CP	A,E	
BB	CP	E	
BC	CP	A,H	
BC	CP	H	
BD	CP	A,L	
BD	CP	L	
BE	CP	(HL)	
BE	CP	A,(HL)	
BF	CP	A	
BF	CP	A,A	
C0	RET	NZ	X
C1	POP	BC	L
C2 34 12	JP	NZ,1234H	I X
C3 34 12	JP	1234H	I X
C4 34 12	CALL	NZ,1234H	I X
C5	PUSH	BC	L
C6 12	ADD	A,12H	
C7	RST	00H	X
C8	RET	Z	X
C9	RET		X
CA 34 12	JP	Z,1234H	I X
CB 00	RLC	B	
CB 01	RLC	C	
CB 02	RLC	D	
CB 03	RLC	E	
CB 04	RLC	H	
CB 05	RLC	L	
CB 06	RLC	(HL)	
CB 07	RLC	A	
CB 08	RRC	B	
CB 09	RRC	C	
CB 0A	RRC	D	
CB 0B	RRC	E	
CB 0C	RRC	H	
CB 0D	RRC	L	
CB 0E	RRC	(HL)	
CB 0F	RRC	A	
CB 10	RL	B	
CB 11	RL	C	
CB 12	RL	D	
CB 13	RL	E	
CB 14	RL	H	
CB 15	RL	L	
CB 16	RL	(HL)	
CB 17	RL	A	
CB 18	RR	B	
CB 19	RR	C	
CB 1A	RR	D	
CB 1B	RR	E	
CB 1C	RR	H	
CB 1D	RR	L	
CB 1E	RR	(HL)	
CB 1F	RR	A	
CB 20	SLA	B	
CB 21	SLA	C	
CB 22	SLA	D	
CB 23	SLA	E	
CB 24	SLA	H	
CB 25	SLA	L	
CB 26	SLA	(HL)	
CB 27	SLA	A	
CB 28	SRA	B	
CB 29	SRA	C	
CB 2A	SRA	D	
CB 2B	SRA	E	
CB 2C	SRA	H	
CB 2D	SRA	L	
CB 2E	SRA	(HL)	
CB 2F	SRA	A	
CB 30	EX	B,B'	
CB 31	EX	C,C'	
CB 32	EX	D,D'	
CB 33	EX	E,E'	
CB 34	EX	H,H'	
CB 35	EX	L,L'	
CB 37	EX	A,A'	
CB 38	SRL	B	
CB 39	SRL	C	
CB 3A	SRL	D	
CB 3B	SRL	E	
CB 3C	SRL	H	
CB 3D	SRL	L	
CB 3E	SRL	(HL)	
CB 3F	SRL	A	
CB 40	BIT	0,B	
CB 41	BIT	0,C	
CB 42	BIT	0,D	
CB 43	BIT	0,E	
CB 44	BIT	0,H	
CB 45	BIT	0,L	
CB 46	BIT	0,(HL)	
CB 47	BIT	0,A	
CB 48	BIT	1,B	
CB 49	BIT	1,C	
CB 4A	BIT	1,D	
CB 4B	BIT	1,E	
CB 4C	BIT	1,H	
CB 4D	BIT	1,L	
CB 4E	BIT	1,(HL)	
CB 4F	BIT	1,A	
CB 50	BIT	2,B	

Object Code	Source Code	Mode
CB 51	BIT 2,C	
CB 52	BIT 2,D	
CB 53	BIT 2,E	
CB 54	BIT 2,H	
CB 55	BIT 2,L	
CB 56	BIT 2,(HL)	
CB 57	BIT 2,A	
CB 58	BIT 3,B	
CB 59	BIT 3,C	
CB 5A	BIT 3,D	
CB 5B	BIT 3,E	
CB 5C	BIT 3,H	
CB 5D	BIT 3,L	
CB 5E	BIT 3,(HL)	
CB 5F	BIT 3,A	
CB 60	BIT 4,B	
CB 61	BIT 4,C	
CB 62	BIT 4,D	
CB 63	BIT 4,E	
CB 64	BIT 4,H	
CB 65	BIT 4,L	
CB 66	BIT 4,(HL)	
CB 67	BIT 4,A	
CB 68	BIT 5,B	
CB 69	BIT 5,C	
CB 6A	BIT 5,D	
CB 6B	BIT 5,E	
CB 6C	BIT 5,H	
CB 6D	BIT 5,L	
CB 6E	BIT 5,(HL)	
CB 6F	BIT 5,A	
CB 70	BIT 6,B	
CB 71	BIT 6,C	
CB 72	BIT 6,D	
CB 73	BIT 6,E	
CB 74	BIT 6,H	
CB 75	BIT 6,L	
CB 76	BIT 6,(HL)	
CB 77	BIT 6,A	
CB 78	BIT 7,B	
CB 79	BIT 7,C	
CB 7A	BIT 7,D	
CB 7B	BIT 7,E	
CB 7C	BIT 7,H	
CB 7D	BIT 7,L	
CB 7E	BIT 7,(HL)	
CB 7F	BIT 7,A	
CB 80	RES 0,B	
CB 81	RES 0,C	
CB 82	RES 0,D	
CB 83	RES 0,E	
CB 84	RES 0,H	
CB 85	RES 0,L	
CB 86	RES 0,(HL)	
CB 87	RES 0,A	
CB 88	RES 1,B	
CB 89	RES 1,C	
CB 8A	RES 1,D	
CB 8B	RES 1,E	
CB 8C	RES 1,H	
CB 8D	RES 1,L	
CB 8E	RES 1,(HL)	
CB 8F	RES 1,A	
CB 90	RES 2,B	
CB 91	RES 2,C	
CB 92	RES 2,D	
CB 93	RES 2,E	
CB 94	RES 2,H	
CB 95	RES 2,L	
CB 96	RES 2,(HL)	
CB 97	RES 2,A	
CB 98	RES 3,B	
CB 99	RES 3,C	
CB 9A	RES 3,D	
CB 9B	RES 3,E	
CB 9C	RES 3,H	
CB 9D	RES 3,L	
CB 9E	RES 3,(HL)	
CB 9F	RES 3,A	
CB A0	RES 4,B	
CB A1	RES 4,C	
CB A2	RES 4,D	
CB A3	RES 4,E	
CB A4	RES 4,H	
CB A5	RES 4,L	
CB A6	RES 4,(HL)	
CB A7	RES 4,A	
CB A8	RES 5,B	
CB A9	RES 5,C	
CB AA	RES 5,D	
CB AB	RES 5,E	
CB AC	RES 5,H	
CB AD	RES 5,L	
CB AE	RES 5,(HL)	
CB AF	RES 5,A	
CB B0	RES 6,B	
CB B1	RES 6,C	
CB B2	RES 6,D	
CB B3	RES 6,E	
CB B4	RES 6,H	
CB B5	RES 6,L	
CB B6	RES 6,(HL)	
CB B7	RES 6,A	
CB B8	RES 7,B	
CB B9	RES 7,C	
CB BA	RES 7,D	
CB BB	RES 7,E	
CB BC	RES 7,H	

Object Code	Source Code	Mode
CB BD	RES 7,L	
CB BE	RES 7,(HL)	
CB BF	RES 7,A	
CB C0	SET 0,B	
CB C1	SET 0,C	
CB C2	SET 0,D	
CB C3	SET 0,E	
CB C4	SET 0,H	
CB C5	SET 0,L	
CB C6	SET 0,(HL)	
CB C7	SET 0,A	
CB C8	SET 1,B	
CB C9	SET 1,C	
CB CA	SET 1,D	
CB CB	SET 1,E	
CB CC	SET 1,H	
CB CD	SET 1,L	
CB CE	SET 1,(HL)	
CB CF	SET 1,A	
CB D0	SET 2,B	
CB D1	SET 2,C	
CB D2	SET 2,D	
CB D3	SET 2,E	
CB D4	SET 2,H	
CB D5	SET 2,L	
CB D6	SET 2,(HL)	
CB D7	SET 2,A	
CB D8	SET 3,B	
CB D9	SET 3,C	
CB DA	SET 3,D	
CB DB	SET 3,E	
CB DC	SET 3,H	
CB DD	SET 3,L	
CB DE	SET 3,(HL)	
CB DF	SET 3,A	
CB E0	SET 4,B	
CB E1	SET 4,C	
CB E2	SET 4,D	
CB E3	SET 4,E	
CB E4	SET 4,H	
CB E5	SET 4,L	
CB E6	SET 4,(HL)	
CB E7	SET 4,A	
CB E8	SET 5,B	
CB E9	SET 5,C	
CB EA	SET 5,D	
CB EB	SET 5,E	
CB EC	SET 5,H	
CB ED	SET 5,L	
CB EE	SET 5,(HL)	
CB EF	SET 5,A	
CB F0	SET 6,B	
CB F1	SET 6,C	
CB F2	SET 6,D	

Object Code	Source Code	Mode		
CB F3	SET 6,E			
CB F4	SET 6,H			
CB F5	SET 6,L			
CB F6	SET 6,(HL)			
CB F7	SET 6,A			
CB F8	SET 7,B			
CB F9	SET 7,C			
CB FA	SET 7,D			
CB FB	SET 7,E			
CB FC	SET 7,H			
CB FD	SET 7,L			
CB FE	SET 7,(HL)			
CB FF	SET 7,A			
CC 34 12	CALL Z,1234H	I	X	
CD 34 12	CALL 1234H	I	X	
CE 12	ADD A,12H			
CF	RST 08H		X	
D0	RET NC		X	
D1	POP DE			L
D2 34 12	JP NC,1234H	I	X	
D3 12	OUT (12H),A			
D4 34 12	CALL NC,1234H	I	X	
D5	PUSH DE			L
D6 12	SUB 12H			
D6 12	SUB A,12H			
D7	RST 10H		X	
D8	RET C		X	
D9	EXX			
DA 34 12	JP C,1234H	I	X	
DB 12	IN A,(12H)			
DC 34 12	CALL C,1234H	I	X	
DD 01	LD (BC),IX			L
DD 02	LD BC,DE			L
DD 03	LD IX,(BC)			L
DD 07	LD IX,BC			L
DD 09	ADD IX,BC		X	
DD 0B	LD BC,IX			L
DD 0C	LD BC,(BC)			L
DD 0D	LD BC,(DE)			L
DD 0F	LD BC,(HL)			L
DD 10 34 12	DJNZ 1234H		X	
DD 11	LD (DE),IX			L
DD 12	LD DE,DE			L
DD 13	LD IX,(DE)			L
DD 17	LD IX,DE			L
DD 18 34 12	JR 1234H		X	
DD 19	ADD IX,DE		X	
DD 1B	LD DE,IX			L
DD 1C	LD DE,(BC)			L
DD 1D	LD DE,(DE)			L
DD 1F	LD DE,(HL)			L
DD 20 34 12	JR NZ,1234H		X	
DD 21 34 12	LD IX,1234H	I		L
DD 22 34 12	LD (1234H),IX	I		L

Object Code	Source Code	Mode
DD 23	INC IX	X
DD 23	INCW IX	X
DD 24	INC IXU	
DD 25	DEC IXU	
DD 26 12	LD IXU,12H	
DD 27	LD IX,IY	L
DD 28 34 12	JR Z,1234H	X
DD 29	ADD IX,IX	X
DD 2A 34 12	LD IX,(1234H)	I L
DD 2B	DEC IX	X
DD 2B	DECW IX	X
DD 2C	INC IXL	
DD 2D	DEC IXL	
DD 2E 12	LD IXL,12H	
DD 2F	CPLW HL	
DD 2F	CPLW	
DD 30 34 12	JR NC,1234H	X
DD 31	LD (HL),IX	L
DD 32	LD HL,DE	L
DD 33	LD IX,(HL)	L
DD 34 12	INC (IX+12H)	I
DD 35 12	DEC (IX+12H)	I
DD 36 12 34	LD (IX+12H),34H	I
DD 37	LD IX,HL	L
DD 38 34 12	JR C,1234H	X
DD 39	ADD IX,SP	X
DD 3B	LD HL,IX	L
DD 3C	LD HL,(BC)	L
DD 3D	LD HL,(DE)	L
DD 3E	SWAP IX	
DD 3F	LD HL,(HL)	L
DD 40	INW BC,(C)	
DD 41	OUTW (C),BC	
DD 44	LD B,IXU	
DD 45	LD B,IXL	
DD 46 12	LD B,(IX+12H)	I
DD 47	LD I,HL	L
DD 47	LDW I,HL	L
DD 4C	LD C,IXU	
DD 4D	LD C,IXL	
DD 4E 12	LD C,(IX+12H)	I
DD 50	INW DE,(C)	
DD 51	OUTW (C),DE	
DD 54	LD D,IXU	
DD 55	LD D,IXL	
DD 56 12	LD D,(IX+12H)	I
DD 57	LD HL,I	L
DD 57	LDW HL,I	L
DD 5D	LD E,IXL	
DD 5D	LD E,IYL	
DD 5E 12	LD E,(IX+12H)	I
DD 60	LD IXU,B	
DD 61	LD IXU,C	
DD 62	LD IXU,D	

Object Code	Source Code	Mode
DD 63	LD IXU,E	
DD 64	LD IXU,IXU	
DD 65	LD IXU,IXL	
DD 66 12	LD H,(IX+12H)	I
DD 67	LD IXU,A	
DD 68	LD IXL,B	
DD 69	LD IXL,C	
DD 6A	LD IXL,D	
DD 6B	LD IXL,E	
DD 6C	LD IXL,IXU	
DD 6D	LD IXL,IXL	
DD 6E 12	LD L,(IX+12H)	I
DD 6F	LD IXL,A	
DD 70 12	LD (IX+12H),B	I
DD 71 12	LD (IX+12H),C	I
DD 72 12	LD (IX+12H),D	I
DD 73 12	LD (IX+12H),E	I
DD 74 12	LD (IX+12H),H	I
DD 75 12	LD (IX+12H),L	I
DD 77 12	LD (IX+12H),A	I
DD 78	INW HL,(C)	
DD 79	OUTW (C),HL	
DD 7C	LD A,IXU	
DD 7D	LD A,IXL	
DD 7E 12	LD A,(IX+12H)	I
DD 84	ADD A,IXU	
DD 85	ADD A,IXL	
DD 86 12	ADD A,(IX+12H)	I
DD 87	ADDW HL,IX	
DD 87	ADDW IX	
DD 8C	ADC A,IXU	
DD 8D	ADC A,IXL	
DD 8E 12	ADC A,(IX+12H)	I
DD 8F	ADCW HL,IX	
DD 8F	ADCW IX	
DD 94	SUB A,IXU	
DD 95	SUB A,IXL	
DD 96 12	SUB A,(IX+12H)	I
DD 97	SUBW HL,IX	
DD 97	SUBW IX	
DD 9C	SBC A,IXU	
DD 9D	SBC A,IXL	
DD 9E 12	SBC A,(IX+12H)	I
DD 9F	SBCW HL,IX	
DD 9F	SBCW IX	
DD A4	AND A,IXU	
DD A4	AND IXU	
DD A5	AND A,IXL	
DD A5	AND IXL	
DD A6 12	AND (IX+12H)	I
DD A6 12	AND A,(IX+12H)	I
DD A7	ANDW HL,IX	
DD A7	ANDW IX	
DD AC	XOR A,IXU	

Object Code	Source Code		Mode		
DD AC	XOR	IXU			
DD AD	XOR	A,IXL			
DD AD	XOR	IXL			
DD AE 12	XOR	(IX+12H)	I		
DD AE 12	XOR	A,(IX+12H)	I		
DD AF	XORW	HL,IX			
DD AF	XORW	IX			
DD B4	OR	A,IXU			
DD B4	OR	IXU			
DD B5	OR	A,IXL			
DD B5	OR	IXL			
DD B6 12	OR	(IX+12H)	I		
DD B6 12	OR	A,(IX+12H)	I		
DD B7	ORW	HL,IX			
DD B7	ORW	IX			
DD BC	CP	A,IXU			
DD BC	CP	IXU			
DD BD	CP	A,IXL			
DD BD	CP	IXL			
DD BE 12	CP	(IX+12H)	I		
DD BE 12	CP	A,(IX+12H)	I		
DD BF	CPW	HL,IX			
DD BF	CPW	IX			
DD C0	DDIR	W			
DD C1	DDIR	IB,W			
DD C2	DDIR	IW,W			
DD C3	DDIR	IB			
DD C4 34 12	CALR	NZ,1234H		X	
DD C6 12	ADDW	(IX+12H)	I		
DD C6 12	ADDW	HL,(IX+12H)	I		
DD C8	LDCTL	SR,A			
DD CA 01	LDCTL	SR,01H			
DD CB 12 01	LD	BC,(SP+12H)	I		L
DD CB 12 02	RLCW	(IX+12H)	I		
DD CB 12 03	LD	BC,(IX+12H)	I		L
DD CB 12 06	RLC	(IX+12H)	I		
DD CB 12 09	LD	(SP+12H),BC	I		L
DD CB 12 0A	RRCW	(IX+12H)	I		
DD CB 12 0B	LD	(IX+12H),BC	I		L
DD CB 12 0E	RRC	(IX+12H)	I		
DD CB 12 11	LD	DE,(SP+12H)	I		L
DD CB 12 12	RLW	(IX+12H)	I		
DD CB 12 13	LD	DE,(IX+12H)	I		L
DD CB 12 16	RL	(IX+12H)	I		
DD CB 12 19	LD	(SP+12H),DE	I		L
DD CB 12 1A	RRW	(IX+12H)	I		
DD CB 12 1B	LD	(IX+12H),DE	I		L
DD CB 12 1E	RR	(IX+12H)	I		
DD CB 12 21	LD	IX,(SP+12H)	I		L
DD CB 12 22	SLAW	(IX+12H)	I		
DD CB 12 23	LD	IY,(IX+12H)	I		L
DD CB 12 26	SLA	(IX+12H)	I		
DD CB 12 29	LD	(SP+12H),IX	I		L
DD CB 12 2A	SRAW	(IX+12H)	I		

Object Code	Source Code		Mode		
DD CB 12 2B	LD	(IX+12H),IY	I		L
DD CB 12 2E	SRA	(IX+12H)	I		
DD CB 12 31	LD	HL,(SP+12H)	I		L
DD CB 12 33	LD	HL,(IX+12H)	I		L
DD CB 12 39	LD	(SP+12H),HL	I		L
DD CB 12 3A	SRLW	(IX+12H)	I		
DD CB 12 3B	LD	(IX+12H),HL	I		L
DD CB 12 3E	SRL	(IX+12H)	I		
DD CB 12 46	BIT	0,(IX+12H)	I		
DD CB 12 4E	BIT	1,(IX+12H)	I		
DD CB 12 56	BIT	2,(IX+12H)	I		
DD CB 12 5E	BIT	3,(IX+12H)	I		
DD CB 12 66	BIT	4,(IX+12H)	I		
DD CB 12 6E	BIT	5,(IX+12H)	I		
DD CB 12 76	BIT	6,(IX+12H)	I		
DD CB 12 7E	BIT	7,(IX+12H)	I		
DD CB 12 86	RES	0,(IX+12H)	I		
DD CB 12 8E	RES	1,(IX+12H)	I		
DD CB 12 92	MULTW	(IX+12H)	I		
DD CB 12 92	MULTW	HL,(IX+12H)	I		
DD CB 12 96	RES	2,(IX+12H)	I		
DD CB 12 9A	MULTUW	(IX+12H)	I		
DD CB 12 9A	MULTUW	HL,(IX+12H)	I		
DD CB 12 9E	RES	3,(IX+12H)	I		
DD CB 12 A6	RES	4,(IX+12H)	I		
DD CB 12 AE	RES	5,(IX+12H)	I		
DD CB 12 B6	RES	6,(IX+12H)	I		
DD CB 12 BA	DIVUW	(IX+12H)	I		
DD CB 12 BA	DIVUW	HL,(IX+12H)	I		
DD CB 12 BE	RES	7,(IX+12H)	I		
DD CB 12 C6	SET	0,(IX+12H)	I		
DD CB 12 CE	SET	1,(IX+12H)	I		
DD CB 12 D6	SET	2,(IX+12H)	I		
DD CB 12 DE	SET	3,(IX+12H)	I		
DD CB 12 E6	SET	4,(IX+12H)	I		
DD CB 12 EE	SET	5,(IX+12H)	I		
DD CB 12 F6	SET	6,(IX+12H)	I		
DD CB 12 FE	SET	7,(IX+12H)	I		
DD CC 34 12	CALR	Z,1234H		X	
DD CD 34 12	CALR	1234H		X	
DD CE 12	ADCW	(IX+12H)	I		
DD CE 12	ADCW	HL,(IX+12H)	I		
DD CF	MTEST				
DD D0	LDCTL	A,XSR			
DD D4 34 12	CALR	NC,1234H		X	
DD D6 12	SUBW	(IX+12H)			
DD D6 12	SUBW	HL,(IX+12H)	I		
DD D8	LDCTL	XSR,A			
DD D9	EXXX				
DD DA 01	LDCTL	XSR,01H			
DD DC 34 12	CALR	C,1234H		X	
DD DE 12	SBCW	(IX+12H)	I		
DD DE 12	SBCW	HL,(IX+12H)			
DD E1	POP	IX			L

Object Code	Source Code		Mode		
DD E3	EX	(SP),IX			L
DD E4 34 12	CALR	PO,1234H		X	
DD E5	PUSH	IX			L
DD E6 12	ANDW	(IX+12H)	I		
DD E6 12	ANDW	HL,(IX+12H)	I		
DD E9	JP	(IX)		X	
DD EC 34 12	CALR	PE,1234H		X	
DD EE 12	XORW	(IX+12H)	I		
DD EE 12	XORW	HL,(IX+12H)	I		
DD F3 1F	DI	1FH			
DD F4 34 12	CALR	P,1234H		X	
DD F6 12	ORW	(IX+12H)	I		
DD F6 12	ORW	HL,(IX+12H)	I		
DD F7	SETC	LW			
DD F9	LD	SP,IX			L
DD FB 1F	EI	1FH			
DD FC 34 12	CALR	M,1234H		X	
DD FE 12	CPW	(IX+12H)	I		
DD FE 12	CPW	HL,(IX+12H)	I		
DD FF	RESC	LW			
DE 12	SBC	A,12H			
DF	RST	18H		X	
E0	RET	NV		X	
E0	RET	PO		X	
E1	POP	HL			L
E2 34 12	JP	NV,1234H	I	X	
E2 34 12	JP	PO,1234H	I	X	
E3	EX	(SP),HL			L
E4 34 12	CALL	NV, 1234H	I	X	
E4 34 12	CALL	PO,1234H	I	X	
E5	PUSH	HL			L
E6 12	AND	12H			
E6 12	AND	A,12H			
E7	RST	20H		X	
E8	RET	PE		X	
E8	RET	V		X	
E9	JP	(HL)		X	
EA 34 12	JP	PE,1234H	I	X	
EA 34 12	JP	V,1234H	I	X	
EB	EX	DE,HL			L
EC 34 12	CALL	V, 1234H	I	X	
EC 34 12	CALL	PE,1234H	I	X	
ED 00 12	IN0	B,(12H)			
ED 01 12	OUT0	(12H),B			
ED 02	LD	BC,BC			L
ED 03	EX	BC,IX			L
ED 04	TST	B			
ED 05	EX	BC,DE			L
ED 06 34 12	LDW	(BC),1234H	I		L
ED 07	EX	A,B			
ED 08 12	IN0	C,(12H)			
ED 09 12	OUT0	(12H),C			
ED 0B	EX	BC,IY			L
ED 0C	TST	C			
ED 0D	EX	BC,HL			L
ED 0E	SWAP	BC			

Object Code	Source Code		Mode		
ED 0F	EX	A,C			
ED 10 12	IN0	D,(12H)			
ED 11 12	OUT0	(12H),D			
ED 12	LD	DE,BC			L
ED 13	EX	DE,IX			L
ED 14	TST	D			
ED 16 34 12	LDW	(DE),1234H	I		L
ED 17	EX	A,D			
ED 18 12	IN0	E,(12H)			
ED 19 12	OUT0	(12H),E			
ED 1B	EX	DE,IY			L
ED 1C	TST	E			
ED 1E	SWAP	DE			
ED 1F	EX	A,E			
ED 20 12	IN0	H,(12H)			
ED 21 12	OUT0	(12H),H			
ED 24	TST	H			
ED 27	EX	A,H			
ED 28 12	IN0	L,(12H)			
ED 29 12	OUT0	(12H),L			
ED 2B	EX	IX,IY			L
ED 2C	TST	L			
ED 2F	EX	A,L			
ED 30 12	IN0	(12H)			
ED 32	LD	HL,BC			L
ED 33	EX	HL,IX			L
ED 34	TST	(HL)			
ED 36 34 12	LDW	(HL),1234H	I		L
ED 37	EX	A,(HL)			
ED 38 12	IN0	A,(12H)			
ED 39 12	OUT0	(12H),A			
ED 3B	EX	HL,IY			L
ED 3C	TST	A			
ED 3E	SWAP	HL			
ED 3F	EX	A,A			
ED 40	IN	B,(C)			
ED 41	OUT	(C),B			
ED 42	SBC	HL,BC			
ED 43 34 12	LD	(1234H),BC	I		L
ED 44	NEG	A			
ED 44	NEG				
ED 45	RETN			X	
ED 46	IM	0			
ED 47	LD	I,A			
ED 48	IN	C,(C)			
ED 49	OUT	(C),C			
ED 4A	ADC	HL,BC			
ED 4B 34 12	LD	BC,(1234H)	I		L
ED 4C	MLT	BC			
ED 4D	RETI			X	
ED 4E	IM	3			
ED 4F	LD	R,A			
ED 50	IN	D,(C)			
ED 51	OUT	(C),D			

Object Code	Source Code	Mode
ED 52	SBC HL,DE	
ED 53 34 12	LD (1234H),DE	I L
ED 54	NEGW HL	
ED 54	NEGW	
ED 55	reserved	
ED 56	IM 1	
ED 57	LD A,I	
ED 58	IN E,(C)	
ED 59	OUT (C),E	
ED 5A	ADC HL,DE	
ED 5B 34 12	LD DE,(1234H)	I L
ED 5C	MLT DE	
ED 5E	IM 2	
ED 5F	LD A,R	
ED 60	IN H,(C)	
ED 61	OUT (C),H	
ED 62	SBC HL,HL	
ED 63 34 12	LD (1234H),HL	I L
ED 64 12	TST 12H	
ED 65	EXTS A	L
ED 65	EXTS	L
ED 67	RRD	
ED 68	IN L,(C)	
ED 69	OUT (C),L	
ED 6A	ADC HL,HL	
ED 6B 34 12	LD HL,(1234H)	I L
ED 6C	MLT HL	
ED 6F	RLD	
ED 71 12	OUT (C),12H	
ED 72	SBC HL,SP	
ED 73 34 12	LD (1234H),SP	I L
ED 74 12	TSTIO 12H	
ED 75	EXTSW HL	
ED 75	EXTSW	
ED 76	SLP	
ED 78	IN A,(C)	
ED 79	OUT (C),A	
ED 7A	ADC HL,SP	
ED 7B 34 12	LD SP,(1234H)	I L
ED 7C	MLT SP	
ED 82 34 12	ADD SP,1234H	I X
ED 83	OTIM	
ED 84	ADDW BC	
ED 84	ADDW HL,BC	
ED 85	ADDW DE	
ED 85	ADDW HL,DE	
ED 86 34 12	ADDW 1234H	
ED 86 34 12	ADDW HL,1234H	
ED 87	ADDW HL	
ED 87	ADDW HL,HL	
ED 8B	OTDM	
ED 8C	ADCW BC	
ED 8C	ADCW HL,BC	
ED 8D	ADCW DE	

Object Code	Source Code	Mode
ED 8D	ADCW HL,DE	
ED 8E 34 12	ADCW 1234H	
ED 8E 34 12	ADCW HL,1234H	
ED 8F	ADCW HL	
ED 8F	ADCW HL,HL	
ED 92 34 12	SUB SP,1234H	I X
ED 93	OTIMR	
ED 94	SUBW BC	
ED 94	SUBW HL,BC	
ED 95	SUBW DE	
ED 95	SUBW HL,DE	
ED 96 34 12	SUBW 1234H	
ED 96 34 12	SUBW HL,1234H	
ED 97	SUBW HL	
ED 97	SUBW HL,HL	
ED 9B	OTDMR	
ED 9C	SBCW BC	
ED 9C	SBCW HL,BC	
ED 9D	SBCW DE	
ED 9D	SBCW HL,DE	
ED 9E 34 12	SBCW 1234H	
ED 9E 34 12	SBCW HL,1234H	
ED 9F	SBCW HL	
ED 9F	SBCW HL,HL	
ED A0	LDI	
ED A1	CPI	X
ED A2	INI	
ED A3	OUTI	
ED A4	ANDW BC	
ED A4	ANDW HL,BC	
ED A5	ANDW DE	
ED A5	ANDW HL,DE	
ED A6 34 12	ANDW 1234H	
ED A6 34 12	ANDW HL,1234H	
ED A7	ANDW HL	
ED A7	ANDW HL,HL	
ED A8	LDD	
ED A9	CPD	X
ED AA	IND	
ED AB	OUTD	
ED AC	XORW BC	
ED AC	XORW HL,BC	
ED AD	XORW DE	
ED AD	XORW HL,DE	
ED AE 34 12	XORW 1234H	
ED AE 34 12	XORW HL,1234H	
ED AF	XORW HL	
ED AF	XORW HL,HL	
ED B0	LDIR	
ED B1	CPIR	X
ED B2	INIR	
ED B3	OTIR	
ED B4	ORW BC	
ED B4	ORW HL,BC	

Object Code	Source Code	Mode
ED B5	ORW DE	
ED B5	ORW HL,DE	
ED B6 34 12	ORW 1234H	
ED B6 34 12	ORW HL,1234H	
ED B7	ORW HL	
ED B7	ORW HL,HL	
ED B8	LDDR	
ED B9	CPDR	X
ED BA	INDR	
ED BB	OTDR	
ED BC	CPW BC	
ED BC	CPW HL,BC	
ED BD	CPW DE	
ED BD	CPW HL,DE	
ED BE 34 12	CPW 1234H	
ED BE 34 12	CPW HL,1234H	
ED BF	CPW HL	
ED BF	CPW HL,HL	
ED C0	LDCTL HL,SR	L
ED C1	POP SR	L
ED C4 12	CALR NZ,12H	X
ED C5	PUSH SR	L
ED C6 34 12	ADD HL,(1234H)	I X
ED C8	LDCTL SR,HL	L
ED CB 00	RLCW BC	
ED CB 01	RLCW DE	
ED CB 02	RLCW (HL)	
ED CB 03	RLCW HL	
ED CB 04	RLCW IX	
ED CB 05	RLCW IY	
ED CB 08	RRCW BC	
ED CB 09	RRCW DE	
ED CB 0A	RRCW (HL)	
ED CB 0B	RRCW HL	
ED CB 0C	RRCW IX	
ED CB 0D	RRCW IY	
ED CB 10	RLW BC	
ED CB 11	RLW DE	
ED CB 12	RLW (HL)	
ED CB 13	RLW HL	
ED CB 14	RLW IX	
ED CB 15	RLW IY	
ED CB 18	RRW BC	
ED CB 19	RRW DE	
ED CB 1A	RRW (HL)	
ED CB 1B	RRW HL	
ED CB 1C	RRW IX	
ED CB 1D	RRW IY	
ED CB 20	SLAW BC	
ED CB 21	SLAW DE	
ED CB 22	SLAW (HL)	
ED CB 23	SLAW HL	
ED CB 24	SLAW IX	
ED CB 25	SLAW IY	

Object Code	Source Code	Mode
ED CB 28	SRAW BC	
ED CB 29	SRAW DE	
ED CB 2A	SRAW (HL)	
ED CB 2B	SRAW HL	
ED CB 2C	SRAW IX	
ED CB 2D	SRAW IY	
ED CB 30	EX BC,BC'	L
ED CB 31	EX DE,DE'	L
ED CB 33	EX HL,HL'	L
ED CB 34	EX IX,IX'	
ED CB 35	EX IY,IY'	L
ED CB 38	SRLW BC	
ED CB 39	SRLW DE	
ED CB 3A	SRLW (HL)	
ED CB 3B	SRLW HL	
ED CB 3C	SRLW IX	
ED CB 3D	SRLW IY	
ED CB 90	MULTW BC	
ED CB 90	MULTW HL,BC	
ED CB 91	MULTW DE	
ED CB 91	MULTW HL,DE	
ED CB 93	MULTW HL	
ED CB 93	MULTW HL,HL	
ED CB 94	MULTW HL,IX	
ED CB 94	MULTW IX	
ED CB 95	MULTW HL,IY	
ED CB 95	MULTW IY	
ED CB 97 34 12	MULTW 1234H	
ED CB 97 34 12	MULTW HL,1234H	
ED CB 98	MULTUW	BC
ED CB 98	MULTUW	HL,BC
ED CB 99	MULTUW	DE
ED CB 99	MULTUW	HL,DE
ED CB 9B	MULTUW	HL
ED CB 9B	MULTUW	HL,HL
ED CB 9C	MULTUW	HL,IX
ED CB 9C	MULTUW	IX
ED CB 9D	MULTUW	HL,IY
ED CB 9D	MULTUW	IY
ED CB 9F	MULTUW	1234H
ED CB 9F	MULTUW	HL,1234H
ED CB B8	DIVUW BC	
ED CB B8	DIVUW HL,BC	
ED CB B9	DIVUW DE	
ED CB B9	DIVUW HL,DE	
ED CB BB	DIVUW HL	
ED CB BB	DIVUW HL,HL	

Object Code	Source Code		Mode		
ED CB BC	DIVUW	HL,IX			
ED CB BC	DIVUW	IX			
ED CB BD	DIVUW	HL,IY			
ED CB BD	DIVUW	IY			
ED CB BF	DIVUW	1234H			
ED CB BF	DIVUW	HL,1234H			
ED CC 12	CALR	Z,12H		X	
ED CD 12	CALR	12H		X	
ED CF	BTEST				
ED D0	LDCTL	A,DSR			
ED D3 34 12	OUTA	(1234H),A	I		
ED D4 12	CALR	NC,12H		X	
ED D6 34 12	SUB	HL,(1234H)	I	X	
ED D8	LDCTL	DSR,A			
ED D9	EXALL				
ED DA 01	LDCTL	DSR,01H			
ED DB 34 12	INA	A,(1234H)	I		
ED DC 12	CALR	C,12H		X	
ED E0	LDIW				L
ED E2	INIW				
ED E3	OUTIW				
ED E4 12	CALR	PO,12H		X	
ED E8	LDDW				L
ED EA	INDW				
ED EB	OUTDW				
ED EC 12	CALR	PE,12H		X	
ED F0	LDIRW				L
ED F2	INIRW				
ED F3	OTIRW				
ED F4 12	CALR	P,12H		X	
ED F7	SETC	LCK			
ED F8	LDDRW				L
ED FA	INDRW				
ED FB	OTDRW				
ED FC 12	CALR	M,12H		X	
ED FF	RESC	LCK			
EE 12	XOR	12H			
EE 12	XOR	A,12H			
EF	RST	28H		X	
F0	RET	NS		X	
F0	RET	P		X	
F1	POP	AF			L
F2 34 12	JP	NS,1234H	I	X	
F2 34 12	JP	P,1234H	I	X	
F3	DI				
F4 34 12	CALL NS	P,1234H	I	X	
F5	PUSH	AF			L
F6 12	OR	12H			
F6 12	OR	A,12H			
F7	RST	30H		X	
F8	RET	M		X	
F8	RET	S		X	
F9	LD	SP,HL			L
FA 34 12	JP	M,1234H	I	X	
FA 34 12	JP	S,1234H	I	X	
FB	EI				
FC 34 12	CALL S,	M,1234H	I	X	
FD 01	LD	(BC),IY			L
FD 02	LD	BC,HL			L
FD 03	LD	IY,(BC)			L
FD 07	LD	IY,BC			L
FD 09	ADD	IY,BC		X	
FD 0B	LD	BC,IY			L
FD 0C	LD	(BC),BC			L
FD 0D	LD	(DE),BC			L
FD 0F	LD	(HL),BC			L
FD 10 56 34 12	DJNZ	123456H		X	
FD 11	LD	(DE),IY			L
FD 12	LD	DE,HL			L
FD 13	LD	IY,(DE)			L
FD 17	LD	IY,DE			L
FD 18 56 34 12	JR	123456H		X	
FD 19	ADD	IY,DE		X	
FD 1B	LD	DE,IY			L
FD 1C	LD	(BC),DE			L
FD 1D	LD	(DE),DE			L
FD 1F	LD	(HL),DE			L
FD 20 56 34 12	JR	NZ,123456H		X	
FD 21 34 12	LD	IY,1234H	I		L
FD 22 34 12	LD	(1234H),IY	I		L
FD 23	INC	IY		X	
FD 23	INCW	IY		X	
FD 24	INC	IYU			
FD 25	DEC	IYU			
FD 27	LD	IY,IX			L
FD 28 56 34 12	JR	Z,123456H		X	
FD 29	ADD	IY,IY		X	
FD 2A 34 12	LD	IY,(1234H)	I		L
FD 2B	DEC	IY		X	
FD 2B	DECW	IY		X	
FD 2C	INC	IYL			
FD 2D	DEC	IYL			
FD 2E 12	LD	IYL,12H			
FD 30 56 34 12	JR	NC,123456H		X	
FD 31	LD	(HL),IY			L
FD 32	LD	HL,HL			L
FD 33	LD	IY,(HL)			L
FD 34 12	INC	(IY+12H)	I		
FD 35 12	DEC	(IY+12H)	I		
FD 36 34 12	LD	(IY+12H),34H	I		
FD 36 12	LD	IYU,12H			
FD 37	LD	IY,HL			L
FD 38 56 34 12	JR	C,123456H		X	
FD 39	ADD	IY,SP		X	
FD 3B	LD	HL,IY			L
FD 3C	LD	(BC),HL			L
FD 3D	LD	(DE),HL			L
FD 3E	SWAP	IY			

Object Code	Source Code		Mode	
FD 3F	LD	(HL),HL		L
FD 44	LD	B,IYU		
FD 45	LD	B,IYL		
FD 46 12	LD	B,(IY+12H)	I	
FD 4C	LD	C,IYU		
FD 4D	LD	C,IYL		
FD 4E 12	LD	C,(IY+12H)	I	
FD 54	LD	D,IYU		
FD 55	LD	D,IYL		
FD 56 12	LD	D,(IY+12H)	I	
FD 5C	LD	E,IYU		
FD 5D	LD	E,IYL		
FD 5E 12	LD	E,(IY+12H)	I	
FD 60	LD	IYU,B		
FD 61	LD	IYU,C		
FD 62	LD	IYU,D		
FD 63	LD	IYU,E		
FD 64	LD	IYU,IYU		
FD 65	LD	IYU,IYL		
FD 66 12	LD	H,(IY+12H)	I	
FD 67	LD	IYU,A		
FD 68	LD	IYL,B		
FD 69	LD	IYL,C		
FD 6A	LD	IYL,D		
FD 6B	LD	IYL,E		
FD 6C	LD	IYL,IYU		
FD 6D	LD	IYL,IYL		
FD 6E 12	LD	L,(IY+12H)	I	
FD 6F	LD	IYL,A		
FD 70 12	LD	(IY+12H),B	I	
FD 71 12	LD	(IY+12H),C	I	
FD 72 12	LD	(IY+12H),D	I	
FD 73 12	LD	(IY+12H),E	I	L
FD 74 12	LD	(IY+12H),H	I	
FD 75 12	LD	(IY+12H),L	I	
FD 77 12	LD	(IY+12H),A	I	
FD 79 34 12	OUTW	(C),1234H		
FD 7C	LD	A,IYU		
FD 7D	LD	A,IYL		
FD 7E 12	LD	A,(IY+12H)	I	
FD 84	ADD	A,IYU		
FD 85	ADD	A,IYL		
FD 86 12	ADD	A,(IY+12H)	I	
FD 87	ADDW	HL,IY		
FD 87	ADDW	IY		
FD 8C	ADC	A,IYU		
FD 8D	ADC	A,IYL		
FD 8E 12	ADC	A,(IY+12H)	I	
FD 8F	ADCW	HL,IY		
FD 8F	ADCW	IY		
FD 94	SUB	A,IYU		
FD 95	SUB	A,IYL		
FD 96 12	SUB	A,(IY+12H)	I	
FD 97	SUBW	HL,IY		
FD 97	SUBW	IY		
FD 9C	SBC	A,IYU		
FD 9D	SBC	A,IYL		
FD 9E 12	SBC	A,(IY+12H)	I	
FD 9F	SBCW	HL,IY		
FD 9F	SBCW	IY		
FD A4	AND	A,IYU		
FD A4	AND	IYU		
FD A5	AND	A,IYL		
FD A5	AND	IYL		
FD A6 12	AND	(IY+12H)	I	
FD A6 12	AND	A,(IY+12H)	I	
FD A7	ANDW	HL,IY		
FD A7	ANDW	IY		
FD AC	XOR	A,IYU		
FD AC	XOR	IYU		
FD AD	XOR	A,IYL		
FD AD	XOR	IYL		
FD AE 12	XOR	(IY+12H)	I	
FD AE 12	XOR	A,(IY+12H)	I	
FD AF	XORW	HL,IY		
FD AF	XORW	IY		
FD B4	OR	A,IYU		
FD B4	OR	IYU		
FD B5	OR	A,IYL		
FD B5	OR	IYL		
FD B6 12	OR	(IY+12H)	I	
FD B6 12	OR	A,(IY+12H)	I	
FD B7	ORW	HL,IY		
FD B7	ORW	IY		
FD BC	CP	A,IYU		
FD BC	CP	IYU		
FD BD	CP	A,IYL		
FD BD	CP	IYL		
FD BE 12	CP	(IY+12H)	I	
FD BE 12	CP	A,(IY+12H)	I	
FD BF	CPW	HL,IY		
FD BF	CPW	IY		
FD C0	DDIR	LW		
FD C1	DDIR	IB,LW		
FD C2	DDIR	IW,LW		
FD C3	DDIR	IW		
FD C4 56 34 12	CALR	NZ,123456H	X	
FD C6 12	ADDW	(IY+12H)	I	
FD C6 12	ADDW	HL,(IY+12H)	I	
FD CB 12 02	RLCW	(IY+12H)	I	
FD CB 12 03	LD	BC,(IY+12H)	I	L
FD CB 12 06	RLC	(IY+12H)	I	
FD CB 12 0A	RRCW	(IY+12H)	I	
FD CB 12 0B	LD	(IY+12H),BC	I	L
FD CB 12 0E	RRC	(IY+12H)	I	
FD CB 12 12	RLW	(IY+12H)	I	
FD CB 12 13	LD	DE,(IY+12H)	I	L
FD CB 12 16	RL	(IY+12H)	I	

Object Code	Source Code		Mode		
FD CB 12 1A	RRW	(IY+12H)	I		
FD CB 12 1B	LD	(IY+12H),DE	I		
FD CB 12 1E	RR	(IY+12H)	I		
FD CB 12 21	LD	IY,(SP+12H)	I		L
FD CB 12 22	SLAW	(IY+12H)	I		
FD CB 12 23	LD	IX,(IY+12H)	I		L
FD CB 12 26	SLA	(IY+12H)	I		
FD CB 12 29	LD	(SP+12H),IY	I		L
FD CB 12 2A	SRAW	(IY+12H)	I		
FD CB 12 2B	LD	(IY+12H),IX	I		L
FD CB 12 2E	SRA	(IY+12H)	I		
FD CB 12 33	LD	HL,(IY+12H)	I		L
FD CB 12 3A	SRLW	(IY+12H)	I		
FD CB 12 3B	LD	(IY+12H),HL	I		L
FD CB 12 3E	SRL	(IY+12H)	I		
FD CB 12 46	BIT	0,(IY+12H)	I		
FD CB 12 4E	BIT	1,(IY+12H)	I		
FD CB 12 56	BIT	2,(IY+12H)	I		
FD CB 12 5E	BIT	3,(IY+12H)	I		
FD CB 12 66	BIT	4,(IY+12H)	I		
FD CB 12 6E	BIT	5,(IY+12H)	I		
FD CB 12 76	BIT	6,(IY+12H)	I		
FD CB 12 7E	BIT	7,(IY+12H)	I		
FD CB 12 86	RES	0,(IY+12H)	I		
FD CB 12 8E	RES	1,(IY+12H)	I		
FD CB 12 92	MULTW	(IY+12H)	I		
FD CB 12 92	MULTW	HL,(IY+12H)	I		
FD CB 12 96	RES	2,(IY+12H)	I		
FD CB 12 9A	MULTUW	(IY+12H)	I		
FD CB 12 9A	MULTUW	HL,(IY+12H)	I		
FD CB 12 9E	RES	3,(IY+12H)	I		
FD CB 12 A6	RES	4,(IY+12H)	I		
FD CB 12 AE	RES	5,(IY+12H)	I		
FD CB 12 B6	RES	6,(IY+12H)	I		
FD CB 12 BA	DIVUW	(IY+12H)	I		
FD CB 12 BA	DIVUW	HL,(IY+12H)	I		
FD CB 12 BE	RES	7,(IY+12H)	I		
FD CB 12 C6	SET	0,(IY+12H)	I		
FD CB 12 CE	SET	1,(IY+12H)	I		
FD CB 12 D6	SET	2,(IY+12H)	I		
FD CB 12 DE	SET	3,(IY+12H)	I		
FD CB 12 E6	SET	4,(IY+12H)	I		
FD CB 12 EE	SET	5,(IY+12H)	I		
FD CB 12 F6	SET	6,(IY+12H)	I		
FD CB 12 FE	SET	7,(IY+12H)	I		
FD CC 56 34 12	CALR	Z,123456H		X	
FD CD 56 34 12	CALR	123456H		X	
FD CE 12	ADCW	(IY+12H)	I		
FD CE 12	ADCW	HL,(IY+12H)	I		
FD D0	LDCTL	A,YSR			
FD D3 34 12	OUTAW	(1234H),HL	I		
FD D4 56 34 12	CALR	NC,123456H		X	
FD D6 12	SUBW	(IY+12H)			
FD D6 12	SUBW	HL,(IY+12H)	I		

Object Code	Source Code		Mode		
FD D8	LDCTL	YSR,A			
FD D9	EXXY				
FD DA 01	LDCTL	YSR,01H			
FD DB 34 12	INAW	HL,(1234H)	I		
FD DC 56 34 12	CALR	C,123456H		X	
FD DE 12	SBCW	(IY+12H)	I		
FD DE 12	SBCW	HL,(IY+12H)			
FD E1	POP	IY			L
FD E3	EX	(SP),IY			L
FD E4 56 34 12	CALR	PO,123456H		X	
FD E5	PUSH	IY			L
FD E6 12	ANDW	(IY+12H)	I		
FD E6 12	ANDW	HL,(IY+12H)	I		
FD E9	JP	(IY)		X	
FD EC 56 34 12	CALR	PE,123456H		X	
FD EE 12	XORW	(IY+12H)	I		
FD EE 12	XORW	HL,(IY+12H)	I		
FD F4 56 34 12	CALR	P,123456H		X	
FD F5 34 12	PUSH	1234H	I		L
FD F6 12	ORW	(IY+12H)	I		
FD F6 12	ORW	HL,(IY+12H)	I		
FD F7	SETC	XM			
FD F9	LD	SP,IY			L
FD FC	CALR	M,123456H		X	
FD FE 12	CPW	(IY+12H)	I		
FD FE 12	CPW	HL,(IY+12H)	I		
FE 12	CP	12H			
FE 12	CP	A,12H			
FF	RST	38H		X	

ZiLOG

Appendix A A

Appendix B B

Appendix C C

Appendix D D

Appendix E E

Index I

Superintegration™ Products Guide S

Zilog's Literature Guide Ordering Information L

APPENDIX D

INSTRUCTIONS AFFECTED BY NORMAL/ EXTENDED MODE, AND LONG WORD MODE

This Appendix has two sets of tables. Each table is a subset of the Table in the Appendix B. The Table D-1 has the instructions which works differently in the Native and Extended mode of operation, and the Table D-2 has the instructions which works differently in Word/Long Word mode of operation.

Table D-1. Instructions operating differently in Native or Extended mode of operation.

Source Code		Object Code				
ADD	HL,BC	09				
ADD	HL,DE	19				
ADD	HL,HL	29				
ADD	HL,SP	39				
ADD	IX,BC	DD	09			
ADD	IX,DE	DD	19			
ADD	IX,IX	DD	29			
ADD	IX,SP	DD	39			
ADD	IY,BC	FD	09			
ADD	IY,DE	FD	19			
ADD	IY,IY	FD	29			
ADD	IY,SP	FD	39			
CALR	123456H	FD	CD	56	34	12
CALR	1234H	DD	CD	34	12	
CALR	12H	ED	CD	12		
CALR	C,123456H	FD	DC	56	34	12
CALR	C,1234H	DD	DC	34	12	
CALR	C,12H	ED	DC	12		
CALR	M,123456H	FD	FC			
CALR	M,1234H	DD	FC	34	12	
CALR	M,12H	ED	FC	12		
CALR	NC,123456H	FD	D4	56	34	12
CALR	NC,1234H	DD	D4	34	12	
CALR	NC,12H	ED	D4	12		
CALR	NZ,123456H	FD	C4	56	34	12
CALR	NZ,1234H	DD	C4	34	12	
CALR	NZ,12H	ED	C4	12		
CALR	P,123456H	FD	F4	56	34	12
CALR	P,1234H	DD	F4	34	12	
CALR	P,12H	ED	F4	12		
CALR	PE,123456H	FD	EC	56	34	12
CALR	PE,1234H	DD	EC	34	12	
CALR	PE,12H	ED	EC	12		
CALR	PO,123456H	FD	E4	56	34	12
CALR	PO,1234H	DD	E4	34	12	
CALR	PO,12H	ED	E4	12		
CALR	Z,123456H	FD	CC	56	34	12
CALR	Z,1234H	DD	CC	34	12	
CALR	Z,12H	ED	CC	12		
CPD		ED	A9			
CPDR		ED	B9			
CPI		ED	A1			
CPIR		ED	B1			
DEC	BC	0B				
DEC	DE	1B				
DEC	HL	2B				
DEC	IX	DD	2B			
DEC	IY	FD	2B			
DEC	SP	3B				
DECW	BC	0B				
DECW	DE	1B				
DECW	HL	2B				
DECW	IX	DD	2B			
DECW	IY	FD	2B			
DECW	SP	3B				
DJNZ	123456H	FD	10	56	34	12
DJNZ	1234H	DD	10	34	12	
DJNZ	12H	10	12			
INC	BC	03				
INC	DE	13				
INC	HL	23				
INC	IX	DD	23			
INC	IY	FD	23			
INC	SP	33				
INCW	BC	03				
INCW	DE	13				
INCW	HL	23				
INCW	IX	DD	23			
INCW	IY	FD	23			
INCW	SP	33				
JP	(HL)	E9				
JP	(IX)	DD	E9			
JP	(IY)	FD	E9			
JR	123456H	FD	18			
JR	1234H	DD	18	34	12	
JR	12H	18	12			
JR	C,123456H	FD	38	56	34	12
JR	C,1234H	DD	38	34	12	
JR	C,12H	38	12			
JR	NC,123456H	FD	30	56	34	12
JR	NC,1234H	DD	30	34	12	
JR	NZ,123456H	FD	20	56	34	12
JR	NZ,1234H	DD	20	34	12	
JR	NZ,12H	20	12			
JR	Z,123456H	FD	28	56	34	12
JR	Z,1234H	DD	28	34	12	
JR	Z,12H	28	12			
RET	C	D8				
RET	M	F8				
RET	NC	D0				
RET	NS	F0				
RET	NV	E0				
RET	NZ	C0				
RET	P	F0				
RET	PE	E8				
RET	PO	E0				
RET	S	F8				
RET	V	E8				
RET	Z	C8				
RET		C9				
RETI		ED	4D			

Source Code	Object Code
RETN	ED 45
RST 00H	C7
RST 08H	CF
RST 10H	D7
RST 18H	DF
RST 20H	E7
RST 28H	EF
RST 30H	F7
RST 38H	FF

Table D-2. Instructions operates different in Long Word Modes.

Source Code	Object Code
EX (SP),HL	E3
EX (SP),IX	DD E3
EX (SP),IY	FD E3
EX BC,BC'	ED CB 30
EX BC,DE	ED 05
EX BC,HL	ED 0D
EX BC,IX	ED 03
EX BC,IY	ED 0B
EX DE,DE'	ED CB 31
EX DE,HL	EB
EX DE,IX	ED 13
EX DE,IY	ED 1B
EX HL,HL'	ED CB 33
EX HL,IX	ED 33
EX HL,IY	ED 3B
EX IX,IX'	ED CB 34
EX IX,IY	ED 2B
EX IY,IY'	ED CB 35
EXTS A	ED 65
EXTS	ED 65
LD (BC),BC	FD 0C
LD (BC),DE	FD 1C
LD (BC),HL	FD 3C
LD (BC),IX	DD 01
LD (BC),IY	FD 01
LD (DE),BC	FD 0D
LD (DE),DE	FD 1D
LD (DE),HL	FD 3D
LD (DE),IX	DD 11
LD (DE),IY	FD 11
LD (HL),BC	FD 0F
LD (HL),DE	FD 1F
LD (HL),HL	FD 3F
LD (HL),IX	DD 31
LD (HL),IY	FD 31
LD BC,(BC)	DD 0C
LD BC,(DE)	DD 0D
LD BC,(HL)	DD 0F
LD BC,BC	ED 02
LD BC,DE	DD 02
LD BC,HL	FD 02
LD BC,IX	DD 0B
LD BC,IY	FD 0B
LD DE,(BC)	DD 1C
LD DE,(DE)	DD 1D
LD DE,(HL)	DD 1F
LD DE,BC	ED 12
LD DE,DE	DD 12
LD DE,HL	FD 12
LD DE,IX	DD 1B
LD DE,IY	FD 1B
LD HL,(BC)	DD 3C
LD HL,(DE)	DD 3D
LD HL,(HL)	DD 3F
LD HL,BC	ED 32
LD HL,DE	DD 32
LD HL,HL	FD 32
LD HL,I	DD 57
LD HL,IX	DD 3B
LD HL,IY	FD 3B
LD I,HL	DD 47
LD IX,(BC)	DD 03
LD IX,(DE)	DD 13
LD IX,(HL)	DD 33
LD IX,BC	DD 07
LD IX,DE	DD 17
LD IX,HL	DD 37
LD IX,IY	DD 27
LD IY,(BC)	FD 03
LD IY,(DE)	FD 13
LD IY,(HL)	FD 33
LD IY,BC	FD 07
LD IY,DE	FD 17
LD IY,HL	FD 37
LD IY,IX	FD 27
LD SP,HL	F9
LD SP,IX	DD F9
LD SP,IY	FD F9

Source Code		Object Code	
LDCTL	HL,SR	ED	C0
LDCTL	SR,HL	ED	C8
LDDRW		ED	F8
LDDW		ED	E8
LDIRW		ED	F0
LDIW		ED	E0
LDW	HL,I	DD	57
LDW	I,HL	DD	47
POP	AF	F1	
POP	BC	C1	
POP	DE	D1	
POP	HL	E1	
POP	IX	DD	E1
POP	IY	FD	E1
POP	SR	ED	C1
PUSH	AF	F5	
PUSH	BC	C5	
PUSH	DE	D5	
PUSH	HL	E5	
PUSH	IX	DD	E5
PUSH	IY	FD	E5
PUSH	SR	ED	C5

Appendix A A

Appendix B B

Appendix C C

Appendix D D

Appendix E E

Index I

Superintegration™ Products Guide S

Zilog's Literature Guide Ordering Information L

APPENDIX E

INSTRUCTIONS AFFECTED BY DDIR IM INSTRUCTIONS

This Appendix has instructions which can be used with the Decoder Directive(s) Extend Immediate. There are eight tables (E1-E8) which are the subset of the Table A, sorted by the category of the instruction.

Note that the instructions listed here does not have the DDIR Decoder Directive in front of the instructions listed below, and notation used here may be different by the assembler to be used.

Table E-1. Valid with DDIR IB in Extended mode. LW bit status does not affect the operation

ADD	HL,(123456H)	ED	C6	56	34	12
ADD	SP,123456H	ED	82	56	34	12
CALL	123456H	CD	56	34	12	
CALL	C,123456H	DC	56	34	12	
CALL	M,123456H	FC	56	34	12	
CALL	NC,123456H	D4	56	34	12	
CALL	NZ,123456H	C4	56	34	12	
CALL	P,123456H	F4	56	34	12	
CALL	PE,123456H	EC	56	34	12	
CALL	PO,123456H	E4	56	34	12	
CALL	Z,123456H	CC	56	34	12	
JP	123456H	C3	56	34	12	
JP	C,123456H	DA	56	34	12	
JP	M,123456H	FA	56	34	12	
JP	NC,123456H	D2	56	34	12	
JP	NS,123456H	F2	56	34	12	
JP	NV,123456H	E2	56	34	12	
JP	NZ,123456H	C2	56	34	12	
JP	P,123456H	F2	56	34	12	
JP	PE,123456H	EA	56	34	12	
JP	PO,123456H	E2	56	34	12	
JP	S,123456H	FA	56	34	12	
JP	V,123456H	EA	56	34	12	
JP	Z,123456H	CA	56	34	12	
SUB	HL,(123456H)	ED	D6	56	34	12
SUB	SP,123456H	ED	92	56	34	12

Table E-2. Valid with DDIR IB. XM bit status does not affect the operation. Transfer size determined by LW bit. (Either with DDIR IB, DDIR IB,LW or DDIR IB,W)

LD	(123456H),BC	ED	43	56	34	12
LD	(123456H),DE	ED	53	56	34	12
LD	(123456H),HL	22	56	34	12	
LD	(123456H),HL	ED	63	56	34	12
LD	(123456H),IX	DD	22	56	34	12
LD	(123456H),IY	FD	22	56	34	12
LD	(123456H),SP	ED	73	56	34	12
LD	(IX+1234H),BC	DD	CB	34	12	0B
LD	(IX+1234H),DE	DD	CB	34	12	1B
LD	(IX+1234H),HL	DD	CB	34	12	3B
LD	(IX+1234H),IY	DD	CB	34	12	2B
LD	(IY+1234H),BC	FD	CB	34	12	0B
LD	(IY+1234H),E	FD	73	34	12	
LD	(IY+1234H),HL	FD	CB	34	12	3B
LD	(IY+1234H),IX	FD	CB	34	12	2B
LD	(SP+1234H),BC	DD	CB	34	12	09
LD	(SP+1234H),DE	DD	CB	34	12	19
LD	(SP+1234H),HL	DD	CB	34	12	39
LD	(SP+1234H),IX	DD	CB	34	12	29
LD	(SP+1234H),IY	FD	CB	34	12	29
LD	BC,(123456H)	ED	4B	56	34	12
LD	BC,(IX+1234H)	DD	CB	34	12	03
LD	BC,(IY+1234H)	FD	CB	34	12	03
LD	BC,(SP+1234H)	DD	CB	34	12	01
LD	DE,(123456H)	ED	5B	56	34	12
LD	DE,(IX+1234H)	DD	CB	34	12	13
LD	DE,(IY+1234H)	FD	CB	34	12	13
LD	DE,(SP+1234H)	DD	CB	34	12	11
LD	HL,(123456H)	2A	56	34	12	
LD	HL,(123456H)	ED	6B	56	34	12
LD	HL,(IX+1234H)	DD	CB	34	12	33
LD	HL,(IY+1234H)	FD	CB	34	12	33
LD	HL,(SP+1234H)	DD	CB	34	12	31
LD	IX,(123456H)	DD	2A	56	34	12
LD	IX,(IY+1234H)	FD	CB	34	12	23
LD	IX,(SP+1234H)	DD	CB	34	12	21
LD	IY,(123456H)	FD	2A	56	34	12
LD	IY,(IX+1234H)	DD	CB	34	12	23
LD	IY,(SP+1234H)	FD	CB	34	12	21
LD	SP,(123456H)	ED	7B	56	34	12
LDW	(BC),123456H	ED	06	56	34	12
LDW	(DE),123456H	ED	16	56	34	12
LDW	(HL),123456H	ED	36	56	34	12

Table E-3. Valid with DDIR IB in Long Word mode. XM bit status does not affect the operation. (Either with DDIR IB,LW or DDIR IB with LW bit set.)

LD	BC,123456H	01	56	34	12	
LD	DE,123456H	11	56	34	12	
LD	HL,123456H	21	56	34	12	
LD	IX,123456H	DD	21	56	34	12
LD	IY,123456H	FD	21	56	34	12
LD	SP,123456H	31	56	34	12	
PUSH	123456H	FD	F5	56	34	12

Table E-4. Valid with DDIR IB. XM bit nor LW bit status do not affect the operation

ADC	A,(IX+1234H)	DD	8E	34	12	
ADC	A,(IY+1234H)	FD	8E	34	12	
ADCW	(IX+1234H)	DD	CE	34	12	
ADCW	(IY+1234H)	FD	CE	34	12	
ADCW	HL,(IX+1234H)	DD	CE	34	12	
ADCW	HL,(IY+1234H)	FD	CE	34	12	
ADD	A,(IX+1234H)	DD	86	34	12	
ADD	A,(IY+1234H)	FD	86	34	12	
ADDW	(IX+1234H)	DD	C6	34	12	
ADDW	(IY+1234H)	FD	C6	34	12	
ADDW	HL,(IX+1234H)	DD	C6	34	12	
ADDW	HL,(IY+1234H)	FD	C6	34	12	
AND	(IX+1234H)	DD	A6	34	12	
AND	(IY+1234H)	FD	A6	34	12	
AND	A,(IX+1234H)	DD	A6	34	12	
AND	A,(IY+1234H)	FD	A6	34	12	
ANDW	(IX+1234H)	DD	E6	34	12	
ANDW	(IY+1234H)	FD	E6	34	12	
ANDW	HL,(IX+1234H)	DD	E6	34	12	
ANDW	HL,(IY+1234H)	FD	E6	34	12	
BIT	0,(IX+1234H)	DD	CB	34	12	46
BIT	0,(IY+1234H)	FD	CB	34	12	46
BIT	1,(IX+1234H)	DD	CB	34	12	4E
BIT	1,(IY+1234H)	FD	CB	34	12	4E
BIT	2,(IX+1234H)	DD	CB	34	12	56
BIT	2,(IY+1234H)	FD	CB	34	12	56
BIT	3,(IX+1234H)	DD	CB	34	12	5E
BIT	3,(IY+1234H)	FD	CB	34	12	5E
BIT	4,(IX+1234H)	DD	CB	34	12	66
BIT	4,(IY+1234H)	FD	CB	34	12	66
BIT	5,(IX+1234H)	DD	CB	34	12	6E
BIT	5,(IY+1234H)	FD	CB	34	12	6E
BIT	6,(IX+1234H)	DD	CB	34	12	76
BIT	6,(IY+1234H)	FD	CB	34	12	76
BIT	7,(IX+1234H)	DD	CB	34	12	7E
BIT	7,(IY+1234H)	FD	CB	34	12	7E
CP	(IX+1234H)	DD	BE	34	12	
CP	(IY+1234H)	FD	BE	34	12	
CP	A,(IX+1234H)	DD	BE	34	12	
CP	A,(IY+1234H)	FD	BE	34	12	
CPW	(IX+1234H)	DD	FE	34	12	
CPW	(IY+1234H)	FD	FE	34	12	
CPW	HL,(IX+1234H)	DD	FE	34	12	
CPW	HL,(IY+1234H)	FD	FE	34	12	
DEC	(IX+1234H)	DD	35	34	12	
DEC	(IY+1234H)	FD	35	34	12	
DIVUW	(IX+1234H)	DD	CB	34	12	BA
DIVUW	(IY+1234H)	FD	CB	34	12	BA
DIVUW	HL,(IX+1234H)	DD	CB	34	12	BA
DIVUW	HL,(IY+1234H)	FD	CB	34	12	BA
INA	A,(123456H)	ED	DB	34	12	
INAW	HL,(123456H)	FD	DB	34	12	
INC	(IX+1234H)	DD	34	12		
INC	(IY+1234H)	FD	34	12		
LD	(123456H),A	32	56	34	12	
LD	(IX+1234H),56H	DD	36	34	12	56
LD	(IX+1234H),A	DD	77	34	12	
LD	(IX+1234H),B	DD	70	34	12	
LD	(IX+1234H),C	DD	71	34	12	
LD	(IX+1234H),D	DD	72	34	12	
LD	(IX+1234H),E	DD	73	34	12	
LD	(IX+1234H),H	DD	74	34	12	
LD	(IX+1234H),L	DD	75	34	12	
LD	(IY+1234H),56H	FD	36	34	12	56
LD	(IY+1234H),A	FD	77	34	12	
LD	(IY+1234H),B	FD	70	34	12	
LD	(IY+1234H),C	FD	71	34	12	
LD	(IY+1234H),D	FD	72	34	12	
LD	(IY+1234H),DE	FD	CB	34	12	1B
LD	(IY+1234H),H	FD	74	34	12	
LD	(IY+1234H),L	FD	75	34	12	
LD	A,(1234H)	3A	34	34	12	
LD	A,(IX+1234H)	DD	7E	34	12	
LD	A,(IY+1234H)	FD	7E	34	12	
LD	B,(IX+1234H)	DD	46	34	12	
LD	B,(IY+1234H)	FD	46	34	12	
LD	C,(IX+1234H)	DD	4E	34	12	
LD	C,(IY+1234H)	FD	4E	34	12	
LD	D,(IX+1234H)	DD	56	34	12	
LD	D,(IY+1234H)	FD	56	34	12	
LD	E,(IX+1234H)	DD	5E	34	12	
LD	E,(IY+1234H)	FD	5E	34	12	
LD	H,(IX+1234H)	DD	66	34	12	
LD	H,(IY+1234H)	FD	66	34	12	
LD	L,(IX+1234H)	DD	6E	34	12	
LD	L,(IY+1234H)	FD	6E	34	12	
MULTUW	(IX+1234H)	DD	CB	34	12	9A
MULTUW	(IY+1234H)	FD	CB	34	12	9A
MULTUW	HL,(IX+1234H)	DD	CB	34	12	9A
MULTUW	HL,(IY+1234H)	FD	CB	34	12	9A
MULTW	(IX+1234H)	DD	CB	34	12	92
MULTW	(IY+1234H)	FD	CB	34	12	92
MULTW	HL,(IX+1234H)	DD	CB	34	12	92
MULTW	HL,(IY+1234H)	FD	CB	34	12	92
OR	(IX+1234H)	DD	B6	34	12	

OR	(IY+1234H)	FD	B6	34	12	
OR	A,(IX+1234H)	DD	B6	34	12	
OR	A,(IY+1234H)	FD	B6	34	12	
ORW	(IX+1234H)	DD	F6	34	12	
ORW	(IY+1234H)	FD	F6	34	12	
ORW	HL,(IX+1234H)	DD	F6	34	12	
ORW	HL,(IY+1234H)	FD	F6	34	12	
OUTA	(123456H),A	ED	D3	56	34	12
OUTAW	(123456H),HL	FD	D3	56	34	12
RES	0,(IX+1234H)	DD	CB	34	12	86
RES	0,(IY+1234H)	FD	CB	34	12	86
RES	1,(IX+1234H)	DD	CB	34	12	8E
RES	1,(IY+1234H)	FD	CB	34	12	8E
RES	2,(IX+1234H)	DD	CB	34	12	96
RES	2,(IY+1234H)	FD	CB	34	12	96
RES	3,(IX+1234H)	DD	CB	34	12	9E
RES	3,(IY+1234H)	FD	CB	34	12	9E
RES	4,(IX+1234H)	DD	CB	34	12	A6
RES	4,(IY+1234H)	FD	CB	34	12	A6
RES	5,(IX+1234H)	DD	CB	34	12	AE
RES	5,(IY+1234H)	FD	CB	34	12	AE
RES	6,(IX+1234H)	DD	CB	34	12	B6
RES	6,(IY+1234H)	FD	CB	34	12	B6
RES	7,(IX+1234H)	DD	CB	34	12	BE
RES	7,(IY+1234H)	FD	CB	34	12	BE
RL	(IX+1234H)	DD	CB	34	12	16
RL	(IY+1234H)	FD	CB	34	12	16
RLC	(IX+1234H)	DD	CB	34	12	06
RLC	(IY+1234H)	FD	CB	34	12	06
RLCW	(IX+1234H)	DD	CB	34	12	02
RLCW	(IY+1234H)	FD	CB	34	12	02
RLW	(IX+1234H)	DD	CB	34	12	12
RLW	(IY+1234H)	FD	CB	34	12	12
RR	(IX+1234H)	DD	CB	34	12	1E
RR	(IY+1234H)	FD	CB	34	12	1E
RRC	(IX+1234H)	DD	CB	34	12	0E
RRC	(IY+1234H)	FD	CB	34	12	0E
RRCW	(IX+1234H)	DD	CB	34	12	0A
RRCW	(IY+1234H)	FD	CB	34	12	0A
RRW	(IX+1234H)	DD	CB	34	12	1A
RRW	(IY+1234H)	FD	CB	34	12	1A
SBC	A,(IX+1234H)	DD	9E	34	12	
SBC	A,(IY+1234H)	FD	9E	34	12	
SBCW	(IX+1234H)	DD	DE	34	12	
SBCW	(IY+1234H)	FD	DE	34	12	
SET	0,(IX+1234H)	DD	CB	34	12	C6
SET	0,(IY+1234H)	FD	CB	34	12	C6
SET	1,(IX+1234H)	DD	CB	34	12	CE
SET	1,(IY+1234H)	FD	CB	34	12	CE
SET	2,(IX+1234H)	DD	CB	34	12	D6
SET	2,(IY+1234H)	FD	CB	34	12	D6
SET	3,(IX+1234H)	DD	CB	34	12	DE
SET	3,(IY+1234H)	FD	CB	34	12	DE
SET	4,(IX+1234H)	DD	CB	34	12	E6
SET	4,(IY+1234H)	FD	CB	34	12	E6
SET	5,(IX+1234H)	DD	CB	34	12	EE
SET	5,(IY+1234H)	FD	CB	34	12	EE
SET	6,(IX+1234H)	DD	CB	34	12	F6
SET	6,(IY+1234H)	FD	CB	34	12	F6
SET	7,(IX+1234H)	DD	CB	34	12	FE
SET	7,(IY+1234H)	FD	CB	34	12	FE
SLA	(IX+1234H)	DD	CB	34	12	26
SLA	(IY+1234H)	FD	CB	34	12	26
SLAW	(IX+1234H)	DD	CB	34	12	22
SLAW	(IY+1234H)	FD	CB	34	12	22
SRA	(IX+1234H)	DD	CB	34	12	2E
SRA	(IY+1234H)	FD	CB	34	12	2E
SRAW	(IX+1234H)	DD	CB	34	12	2A
SRAW	(IY+1234H)	FD	CB	34	12	2A
SRL	(IX+1234H)	DD	CB	34	12	3E
SRL	(IY+1234H)	FD	CB	34	12	3E
SRLW	(IX+1234H)	DD	CB	34	12	3A
SRLW	(IY+1234H)	FD	CB	34	12	3A
SUB	A,(IX+1234H)	DD	96	34	12	
SUB	A,(IY+1234H)	FD	96	34	12	
SUBW	HL,(IX+1234H)	DD	D6	34	12	
SUBW	HL,(IY+1234H)	FD	D6	34	12	
XOR	(IX+1234H)	DD	AE	34	12	
XOR	(IY+1234H)	FD	AE	34	12	
XOR	A,(IX+1234H)	DD	AE	34	12	
XOR	A,(IY+1234H)	FD	AE	34	12	
XORW	(IX+1234H)	DD	EE	34	12	
XORW	(IY+1234H)	FD	EE	34	12	
XORW	HL,(IX+1234H)	DD	EE	34	12	
XORW	HL,(IY+1234H)	FD	EE	34	12	

Table E-5. Valid with DDIR IW in Exteded mode. LW bit status does not affect the operation

ADD	HL,(12345678H)	ED	C6	78	56	34	12
ADD	SP,12345678H	ED	82	78	56	34	12
CALL	12345678H	CD	78	56	34	12	
CALL	C,12345678H	DC	78	56	34	12	
CALL	M,12345678H	FC	78	56	34	12	
CALL	NC,12345678H	D4	78	56	34	12	
CALL	NZ,12345678H	C4	78	56	34	12	
CALL	P,12345678H	F4	78	56	34	12	
CALL	PE,12345678H	EC	78	56	34	12	
CALL	PO,12345678H	E4	78	56	34	12	
CALL	Z,12345678H	CC	78	56	34	12	
JP	12345678H	C3	78	56	34	12	
JP	C,12345678H	DA	78	56	34	12	
JP	M,12345678H	FA	78	56	34	12	
JP	NC,12345678H	D2	78	56	34	12	
JP	NS,12345678H	F2	78	56	34	12	
JP	NV,12345678H	E2	78	56	34	12	
JP	NZ,12345678H	C2	78	56	34	12	
JP	P,12345678H	F2	78	56	34	12	
JP	PE,12345678H	EA	78	56	34	12	
JP	PO,12345678H	E2	78	56	34	12	
JP	S,12345678H	FA	78	56	34	12	
JP	V,12345678H	EA	78	56	34	12	
JP	Z,12345678H	CA	78	56	34	12	
SUB	HL,(12345678H)	ED	D6	78	56	34	12
SUB	SP,12345678H	ED	92	78	56	34	12

Table E-6. Valid with DDIR IW. XM bit status does not affect the operation. Transfer size determined by LW bit

LD	(12345678H),BC	ED	43	78	56	34	12
LD	(12345678H),DE	ED	53	78	56	34	12
LD	(12345678H),HL	22	78	56	34	12	
LD	(12345678H),HL	ED	63	78	56	34	12
LD	(12345678H),IX	DD	22	78	56	34	12
LD	(12345678H),IY	FD	22	78	56	34	12
LD	(12345678H),SP	ED	73	78	56	34	12
LD	(IX+123456H),BC	DD	CB	56	34	12	0B
LD	(IX+123456H),DE	DD	CB	56	34	12	1B
LD	(IX+123456H),HL	DD	CB	56	34	12	3B
LD	(IX+123456H),IY	DD	CB	56	34	12	2B
LD	(IY+123456H),BC	FD	CB	56	34	12	0B
LD	(IY+123456H),E	FD	73	56	34	12	
LD	(IY+123456H),HL	FD	CB	56	34	12	3B
LD	(IY+123456H),IX	FD	CB	56	34	12	2B
LD	(SP+123456H),BC	DD	CB	56	34	12	09
LD	(SP+123456H),DE	DD	CB	56	34	12	19
LD	(SP+123456H),HL	DD	CB	56	34	12	39
LD	(SP+123456H),IX	DD	CB	56	34	12	29
LD	(SP+123456H),IY	FD	CB	56	34	12	29
LD	BC,(12345678H)	ED	4B	78	56	34	12
LD	BC,(IX+123456H)	DD	CB	34	12	03	
LD	BC,(IY+123456H)	FD	CB	34	12	03	
LD	BC,(SP+123456H)	DD	CB	34	12	01	
LD	DE,(12345678H)	ED	5B	78	56	34	12
LD	DE,(IX+123456H)	DD	CB	56	34	12	13
LD	DE,(IY+123456H)	FD	CB	56	34	12	13
LD	DE,(SP+123456H)	DD	CB	56	34	12	11
LD	HL,(12345678H)	2A	78	56	34	12	
LD	HL,(12345678H)	ED	6B	78	56	34	12
LD	HL,(IX+123456H)	DD	CB	56	34	12	33
LD	HL,(IY+123456H)	FD	CB	56	34	12	33
LD	HL,(SP+123456H)	DD	CB	56	34	12	31
LD	IX,(12345678H)	DD	2A	78	56	34	12
LD	IX,(IY+123456H)	FD	CB	56	34	12	23
LD	IX,(SP+123456H)	DD	CB	56	34	12	21
LD	IY,(12345678H)	FD	2A	78	56	34	12
LD	IY,(IX+123456H)	DD	CB	56	34	12	23
LD	IY,(SP+123456H)	FD	CB	56	34	12	21
LD	SP,(12345678H)	ED	7B	78	56	34	12
LDW	(BC),12345678H	ED	06	78	56	34	12
LDW	(DE),12345678H	ED	16	78	56	34	12
LDW	(HL),12345678H	ED	36	78	56	34	12

Table E-7. Valid with DDIR IW in Long Word mode. XM bit status does not affect the operation. (Either with DDIR IW,LW or DDIR IW with LW bit set.)

LD	BC,12345678H	01	78	56	34	12	
LD	DE,12345678H	11	78	56	34	12	
LD	HL,12345678H	21	78	56	34	12	
LD	IX,12345678H	DD	21	78	56	34	12
LD	IY,12345678H	FD	21	78	56	34	12
LD	SP,12345678H	31	78	56	34	12	
PUSH	12345678H	FD	F5	78	56	34	12

Table E-8. Valid with DDIR IW. XM bit nor LW bit status do not affect the operation

ADC	A,(IX+123456H)	DD	8E	56	34	12	
ADC	A,(IY+123456H)	FD	8E	56	34	12	
ADCW	(IX+123456H)	DD	CE	56	34	12	
ADCW	(IY+123456H)	FD	CE	56	34	12	
ADCW	HL,(IX+123456H)	DD	CE	56	34	12	
ADCW	HL,(IY+123456H)	FD	CE	56	34	12	
ADD	A,(IX+123456H)	DD	86	56	34	12	
ADD	A,(IY+123456H)	FD	86	56	34	12	
ADDW	(IX+123456H)	DD	C6	56	34	12	
ADDW	(IY+123456H)	FD	C6	56	34	12	
ADDW	HL,(IX+123456H)	DD	C6	56	34	12	
ADDW	HL,(IY+123456H)	FD	C6	56	34	12	
AND	(IX+123456H)	DD	A6	56	34	12	
AND	(IY+123456H)	FD	A6	56	34	12	
AND	A,(IX+123456H)	DD	A6	56	34	12	
AND	A,(IY+123456H)	FD	A6	56	34	12	
ANDW	(IX+123456H)	DD	E6	56	34	12	
ANDW	(IY+123456H)	FD	E6	56	34	12	
ANDW	HL,(IX+123456H)	DD	E6	56	34	12	
ANDW	HL,(IY+123456H)	FD	E6	56	34	12	
BIT	0,(IX+123456H)	DD	CB	56	34	12	46
BIT	0,(IY+123456H)	FD	CB	56	34	12	46
BIT	1,(IX+123456H)	DD	CB	56	34	12	4E
BIT	1,(IY+123456H)	FD	CB	56	34	12	4E
BIT	2,(IX+123456H)	DD	CB	56	34	12	56
BIT	2,(IY+123456H)	FD	CB	56	34	12	56
BIT	3,(IX+123456H)	DD	CB	56	34	12	5E
BIT	3,(IY+123456H)	FD	CB	56	34	12	5E
BIT	4,(IX+123456H)	DD	CB	56	34	12	66
BIT	4,(IY+123456H)	FD	CB	56	34	12	66
BIT	5,(IX+123456H)	DD	CB	56	34	12	6E
BIT	5,(IY+123456H)	FD	CB	56	34	12	6E
BIT	6,(IX+123456H)	DD	CB	56	34	12	76
BIT	6,(IY+123456H)	FD	CB	56	34	12	76
BIT	7,(IX+123456H)	DD	CB	56	34	12	7E
BIT	7,(IY+123456H)	FD	CB	56	34	12	7E
CP	(IX+123456H)	DD	BE	56	34	12	
CP	(IY+123456H)	FD	BE	56	34	12	
CP	A,(IX+123456H)	DD	BE	56	34	12	
CP	A,(IY+123456H)	FD	BE	56	34	12	
CPW	(IX+123456H)	DD	FE	56	34	12	
CPW	(IY+123456H)	FD	FE	56	34	12	
CPW	HL,(IX+123456H)	DD	FE	56	34	12	
CPW	HL,(IY+123456H)	FD	FE	56	34	12	
DEC	(IX+123456H)	DD	35	56	34	12	
DEC	(IY+123456H)	FD	35	56	34	12	
DIVUW	(IX+123456H)	DD	CB	56	34	12	BA
DIVUW	(IY+123456H)	FD	CB	56	34	12	BA
DIVUW	HL,(IX+123456H)	DD	CB	56	34	12	BA
DIVUW	HL,(IY+123456H)	FD	CB	56	34	12	BA
INA	A,(123456H)	ED	DB	56	34	12	
INAW	HL,(123456H)	FD	DB	56	34	12	
INC	(IX+123456H)	DD	56	34	12		
INC	(IY+123456H)	FD	56	34	12		
LD	(12345678H),A	32	78	56	34	12	
LD	(IX+123456H),56H	DD	36	56	34	12	56
LD	(IX+123456H),A	DD	77	56	34	12	
LD	(IX+123456H),B	DD	70	56	34	12	
LD	(IX+123456H),C	DD	71	56	34	12	
LD	(IX+123456H),D	DD	72	56	34	12	
LD	(IX+123456H),E	DD	73	56	34	12	
LD	(IX+123456H),H	DD	74	56	34	12	
LD	(IX+123456H),L	DD	75	56	34	12	
LD	(IY+123456H),78H	FD	36	56	34	12	78
LD	(IY+123456H),A	FD	77	56	34	12	
LD	(IY+123456H),B	FD	70	56	34	12	
LD	(IY+123456H),C	FD	71	56	34	12	
LD	(IY+123456H),D	FD	72	56	34	12	
LD	(IY+123456H),DE	FD	CB	56	34	12	1B
LD	(IY+123456H),H	FD	74	56	34	12	
LD	(IY+123456H),L	FD	75	56	34	12	
LD	A,(12345678H)	3A	78	56	34	12	
LD	A,(IX+123456H)	DD	7E	56	34	12	
LD	A,(IY+123456H)	FD	7E	56	34	12	
LD	B,(IX+123456H)	DD	46	56	34	12	
LD	B,(IY+123456H)	FD	46	56	34	12	
LD	C,(IX+123456H)	DD	4E	56	34	12	
LD	C,(IY+123456H)	FD	4E	56	34	12	
LD	D,(IX+123456H)	DD	56	56	34	12	
LD	D,(IY+123456H)	FD	56	56	34	12	
LD	E,(IX+123456H)	DD	5E	56	34	12	
LD	E,(IY+123456H)	FD	5E	56	34	12	
LD	H,(IX+123456H)	DD	66	56	34	12	
LD	H,(IY+123456H)	FD	66	56	34	12	
LD	L,(IX+123456H)	DD	6E	56	34	12	
LD	L,(IY+123456H)	FD	6E	56	34	12	
MULTUW	(IX+123456H)	DD	CB	56	34	12	9A
MULTUW	(IY+123456H)	FD	CB	56	34	12	9A
MULTUW	HL,(IX+123456H)	DD	CB	56	34	12	9A
MULTUW	HL,(IY+123456H)	FD	CB	56	34	12	9A
MULTW	(IX+123456H)	DD	CB	56	34	12	92
MULTW	(IY+123456H)	FD	CB	56	34	12	92
MULTW	HL,(IX+123456H)	DD	CB	56	34	12	92
MULTW	HL,(IY+123456H)	FD	CB	56	34	12	92
OR	(IX+123456H)	DD	B6	56	34	12	
OR	(IY+123456H)	FD	B6	56	34	12	

OR	A,(IX+123456H)	DD B6 56 34 12
OR	A,(IY+123456H)	FD B6 56 34 12
ORW	(IX+123456H)	DD F6 56 34 12
ORW	(IY+123456H)	FD F6 56 34 12
ORW	HL,(IX+123456H)	DD F6 56 34 12
ORW	HL,(IY+123456H)	FD F6 56 34 12
OUTA	(12345678H),A	ED D3 78 56 34 12
OUTAW	(12345678H),HL	FD D3 78 56 34 12
RES	0,(IX+123456H)	DD CB 56 34 12 86
RES	0,(IY+123456H)	FD CB 56 34 12 86
RES	1,(IX+123456H)	DD CB 56 34 12 8E
RES	1,(IY+123456H)	FD CB 56 34 12 8E
RES	2,(IX+123456H)	DD CB 56 34 12 96
RES	2,(IY+123456H)	FD CB 56 34 12 96
RES	3,(IX+123456H)	DD CB 56 34 12 9E
RES	3,(IY+123456H)	FD CB 56 34 12 9E
RES	4,(IX+123456H)	DD CB 56 34 12 A6
RES	4,(IY+123456H)	FD CB 56 34 12 A6
RES	5,(IX+123456H)	DD CB 56 34 12 AE
RES	5,(IY+123456H)	FD CB 56 34 12 AE
RES	6,(IX+123456H)	DD CB 56 34 12 B6
RES	6,(IY+123456H)	FD CB 56 34 12 B6
RES	7,(IX+123456H)	DD CB 56 34 12 BE
RES	7,(IY+123456H)	FD CB 56 34 12 BE
RL	(IX+123456H)	DD CB 56 34 12 16
RL	(IY+123456H)	FD CB 56 34 12 16
RLC	(IX+123456H)	DD CB 56 34 12 06
RLC	(IY+123456H)	FD CB 56 34 12 06
RLCW	(IX+123456H)	DD CB 56 34 12 02
RLCW	(IY+123456H)	FD CB 56 34 12 02
RLW	(IX+123456H)	DD CB 56 34 12 12
RLW	(IY+123456H)	FD CB 56 34 12 12
RR	(IX+123456H)	DD CB 56 34 12 1E
RR	(IY+123456H)	FD CB 56 34 12 1E
RRC	(IX+123456H)	DD CB 56 34 12 0E
RRC	(IY+123456H)	FD CB 56 34 12 0E
RRCW	(IX+123456H)	DD CB 56 34 12 0A
RRCW	(IY+123456H)	FD CB 56 34 12 0A
RRW	(IX+123456H)	DD CB 56 34 12 1A
RRW	(IY+123456H)	FD CB 56 34 12 1A
SBC	A,(IX+123456H)	DD 9E 56 34 12
SBC	A,(IY+123456H)	FD 9E 56 34 12
SBCW	(IX+123456H)	DD DE 56 34 12
SBCW	(IY+123456H)	FD DE 56 34 12
SET	0,(IX+123456H)	DD CB 56 34 12 C6
SET	0,(IY+123456H)	FD CB 56 34 12 C6
SET	1,(IX+123456H)	DD CB 56 34 12 CE
SET	1,(IY+123456H)	FD CB 56 34 12 CE
SET	2,(IX+123456H)	DD CB 56 34 12 D6
SET	2,(IY+123456H)	FD CB 56 34 12 D6
SET	3,(IX+123456H)	DD CB 56 34 12 DE
SET	3,(IY+123456H)	FD CB 56 34 12 DE
SET	4,(IX+123456H)	DD CB 56 34 12 E6
SET	4,(IY+123456H)	FD CB 56 34 12 E6
SET	5,(IX+123456H)	DD CB 56 34 12 EE
SET	5,(IY+123456H)	FD CB 56 34 12 EE
SET	6,(IX+123456H)	DD CB 56 34 12 F6
SET	6,(IY+123456H)	FD CB 56 34 12 F6
SET	7,(IX+123456H)	DD CB 56 34 12 FE
SET	7,(IY+123456H)	FD CB 56 34 12 FE
SLA	(IX+123456H)	DD CB 56 34 12 26
SLA	(IY+123456H)	FD CB 56 34 12 26
SLAW	(IX+123456H)	DD CB 56 34 12 22
SLAW	(IY+123456H)	FD CB 56 34 12 22
SRA	(IX+123456H)	DD CB 56 34 12 2E
SRA	(IY+123456H)	FD CB 56 34 12 2E
SRAW	(IX+123456H)	DD CB 56 34 12 2A
SRAW	(IY+123456H)	FD CB 56 34 12 2A
SRL	(IX+123456H)	DD CB 56 34 12 3E
SRL	(IY+123456H)	FD CB 56 34 12 3E
SRLW	(IX+123456H)	DD CB 56 34 12 3A
SRLW	(IY+123456H)	FD CB 56 34 12 3A
SUB	A,(IX+123456H)	DD 96 56 34 12
SUB	A,(IY+123456H)	FD 96 56 34 12
SUBW	HL,(IX+123456H)	DD D6 56 34 12
SUBW	HL,(IY+123456H)	FD D6 56 34 12
XOR	(IX+123456H)	DD AE 56 34 12
XOR	(IY+123456H)	FD AE 56 34 12
XOR	A,(IX+123456H)	DD AE 56 34 12
XOR	A,(IY+123456H)	FD AE 56 34 12
XORW	(IX+123456H)	DD EE 56 34 12
XORW	(IY+123456H)	FD EE 56 34 12
XORW	HL,(IX+123456H)	DD EE 56 34 12
XORW	HL,(IY+123456H)	FD EE 56 34 12

Appendix A A

Appendix B B

Appendix C C

Appendix D D

Appendix E E

Index I

Superintegration™ Products Guide S

Zilog's Literature Guide Ordering Information L

INDEX

Symbols

/INT3-/INT0 6-1
/NMI 6-1
/RESET 1-5
8-Bit Load/Exchange Group 5-6
8080 compatible (Mode 0) 1-5

A

ADC Add with Carry (Word) 5-21
Add (Byte) 5-23
Add (Word) 5-24
Add to Stack Pointer (Word) 5-25
Add/Subtract flag 5-2
Address manipulation 3-1
Address space 1-1
Addressing mode A-1
Addressing mode escape byte A-1
Addressing mode escape bytes, addresses A-2
Addressing Modes 1-4,4-1
AF or AF' Register Select 5-5
AND (Byte) 5-27
AND (Word) 5-28
Arithmetic and Logical Group 5-9
Arithmetic Operation 5-10
Assembly language format 4-1
Assigned Vector Base Register 5-15,6-3,6-6
Assigned Vectors Base 6-4

B

Bank Test 5-30
Bank Test instructions 5-16
BC/DE/HL or BC'/DE'/HL' Register Select 5-4
Binary-coded decimal 4-10,5-1
Bit Test 5-29
Block I/O 5-5
Block move 5-5
Block move, block search, and block I/O instruction 6-1
Block search 5-2, 5-5
Block Transfer and Search Group 5-1,5-8
Block transfer 5-2, 5-8
Bus bandwidth 1-5
Byte ordering 3-2
Byte strings 4-10

C

Call and Restart 5-12
Call Relative 5-1, 5-12
Call Relative 5-32
Call, Return, Push, and Pop 4-10
Carry flag 5-2
Carry or borrow operation 4-6
Chip Version ID Register 2-6,5-15
Compare (Byte) 5-34
Compare (Word) 5-35
Compare and Decrement (Byte) 5-36
Compare and Increment (Byte) 5-38
Compare, Decrement and Repeat (Byte) 5-37
Compare, Increment and Repeat (Byte) 5-39
Complement Accumulator 5-40
Complement Carry flag 5-33
Complement HL Register (Word) 5-41
Condition Codes 5-3
Conditional instructions 5-1
Conditional Return instruction 5-12
Context Switching 1-5,1-6
CPU Control Group 5-16
CPU Control Register Space 2-1
CPU Register Space 2-1

D

Data frame 5-7
Data manipulation 3-1
Data Types 4-1, 4-10
DDIR 3-1
DDIR IB Immediate Byte 3-2
DDIR IB,LW Immediate Byte, Long Word Mode 3-2
DDIR IB,W Immediate Byte, Word Mode 3-2
DDIR IW,LW Immediate Word, Long Word Mode 3-2
DDIR IW,W Immediate Word, Word Mode 3-2
DDIR LW 1-3
DDIR LW Long Word Mode 3-2
DDIR W 1-3
DDIR W Word Mode 3-2
Decimal Adjust Accumulator 4-10,5-2, 5-42
Decoder Directive 3- 2, 5-17, 5-43, A-1
Decrement (Byte) 5-44
Decrement (Word) 5-45

Decrement and Jump if Non-Zero ... 5-12
Decrement and Jump if Non-Zero ... 5-48
Dedicated interrupts (Mode 1) ... 1-5
DIR IW Immediate Word ... 3-2
Direct Address ... 1-4, 4-1, 4-3
Disable and Enable Interrupt ... 5-16
Disable Interrupt instruction ... 6-1
Disable Interrupts ... 5-46
Displacement ... 4-6
Divide Unsigned (Word) ... 5-47

E

Enable Interrupt ... 5-49, 6-1
Enhanced vectored interrupt mode (Mode 3) ... 1-5
Escape codes ... A-1
Example of IM mode ... 4-2
Example of R mode ... 4-1
Example of RA mode ... 4-6
Example of SR mode ... 4-8
Exception Conditions ... 1-4
Exchange Accumulator/Flag with Alternate Bank ... 5-50
Exchange Addressing Register with Top of Stack ... 5-51
Exchange All Registers with Alternate Bank ... 5-56
Exchange instruction ... 2-3,5-6
Exchange IX Register with Alternate Bank ... 5-60
Exchange IY Register with Alternate Bank ... 5-61
Exchange Register (Word) ... 5-52
Exchange Register with Alternate Register (Byte) ... 5-53
Exchange Register with Alternate Register (Word) ... 5-54
Exchange Registers with Alternate Bank ... 5-59
Exchange with Accumulator ... 5-55
Exclusive OR (Byte) ... 5-178
Exclusive OR (Word) ... 5-179
EXECUTION TIME ... 5-18
Extend Sign (Byte) ... 5-57
Extend Sign (Word) ... 5-58
Extended Mode ... 5-4
Extended Registers ... 1-1, 3-1
EXTERNAL I/O ADDRESS SPACE ... 2-1, 2-6
External Input/Output Instruction Group ... 5-13
EXTS instruction ... 5-9

F

F register ... 5-17
Fetching ... 6-2
Flag and Accumulator registers ... 2-1
Flag register ... 2-3, 5-1
Format 1 ... A-1
Format 2 ... A-1
Format 3 ... A-1
Format 4 ... A-1

H

Half-Carry flag ... 5-2
HALT and Sleep instructions ... 5-16
Halt ... 5-62
Hardware reset overrides ... 6-1

I

I Extend ... 6-3
I/O addressing space ... 1-4
I/O instructions ... 1-4
IB decoder directive ... 3-2
IEF1, IEF2 ... 6-3
Immediate ... 1-4
Immediate addressing mode ... 4-2
Increment (Byte) ... 5-70
Increment (Word) ... 5-71
Increment and Decrement instructions ... 5-9
Index Registers ... 2-3
Index registers ... 1-1, 2-1
Indexed ... 1-4,4-1
Indexed address ... 4-5
Indirect Register ... 1-4,4-1,4-3
Indirect Register mode ... 4-3
Input (Byte) ... 5-64
Input (from Page 0) ... 5-67
Input (Word) ... 5-65
Input Accumulator ... 5-66
Input and Decrement (Byte) ... 5-72
Input and Decrement (Word) ... 5-73
Input and Increment (Byte) ... 5-76
Input and Increment (Word) ... 5-77
Input Direct from Port Address (Byte) ... 5-68
Input Direct from Port Address (Word) ... 5-69
Input, Decrement and Repeat (Byte) ... 5-74
Input, Decrement and Repeat (Word) ... 5-75
Input, Increment and Repeat (Byte) ... 5-78
Input, Increment and Repeat (Word) ... 5-79
INSTRUCTION EXECUTION AND EXCEPTIONS ... 5-5
INSTRUCTION SET FUNCTIONAL GROUPS ... 5-6
Instruction decoder ... 3-2
Instruction Execution and Interrupts ... 5-5
Instruction Execution and Trap ... 5-5
Instruction fetch/execute ... 1-3
Instruction mnemonic ... 5-17
Internal Data Bus ... 1-1
Internal I/O Instruction Group ... 5-15
Internal I/O locations ... 5-15
Interrupt acknowledge transaction ... 6-2
Interrupt Control ... 6-2
Interrupt Enable Flag ... 5-5
Interrupt Enable Register ... 2-6,5-15,6-1,6-3
Interrupt Flags and Registers ... 6-2
Interrupt Mode ... 5-5
Interrupt Mode Select ... 5-63

Interrupt mode 6-1,6-2, 6-3, 6-5
Interrupt Priority Ranking 6-2
Interrupt Register 2-3
Interrupt Register Extension 6-2
Interrupt service routines 1-3
Interrupt Vectors Mode 6-6
Interrupt vectors 6-2
interrupt return instruction 5-5
Interrupts 6-1
Interrupts, traps, and resets 1-4,6-1
IW decoder directive 3-2
IX Bank Select 5-4
IX or IX' Register Select 5-4
IY Bank Select 5-4
IY or IY' Register Select 5-4

J

JP 5-80
Jump 5-1,5-80
Jump and Call instructions 4-3
Jump Relative 5-1, 5-81

L

Linear Memory Address Space 1-5
Load Accumulator 5-82
Load Accumulator from R or I register 5-16
Load and Decrement (Byte) 5-96
Load and Decrement (Word) 5-97
Load and Increment (Byte) 5-100
Load and Increment (Word) 5-101
Load B register A-1
Load Control Register (Byte) 5-93
Load from Control Register (Word) 5-94
Load from I or R Register (Byte) 5-90
Load I Register (Word) 5-92
Load Immediate (Byte) 5-83
Load Immediate (Word) 5-84
Load into Control Register (Word) 5-95
Load into I or R Register (Byte) 5-91
Load Register (Byte) 5-86
Load Register (Word) 5-87,5-88, A-1
Load Stack Pointer 5-89
Load, Decrement and Repeat (Byte) 5-98
Load, Decrement and Repeat (Word) 5-99
Load, Exchange, SWAP and Push/Pop Group 5-1
Load, Exchange, SWAP, and PUSH/POP Group 5-7
Load, Increment and Repeat (Byte) 5-102
Load, Increment and Repeat (Word) 5-103
load, arithmetic, logical, shift, and rotate 4-10
Load/Exchange Group 5-1
Lock 5-5
Lock/Unlock status 5-5
Logical, signed numeric, or unsigned 4-10
Long Word Mode 5-5
LW decoder directive 3-2

M

Machine language bit A-1
Main Bank Select 5-4
Maskable Interrupt 6-5
Memory Address Space 2-1
Memory Banking scheme 1-5
memory addressing modes 5-9
Mode Test 5-105
Mode Test instructions 5-16
Multiple register banks 1-5
Multiply (Word) 5-106
Multiply Unsigned (Byte) 5-104
Multiply Unsigned (Word) 5-107

N

NATIVE MODE AND EXTENDED MODE 3-2
Native or Extended mode 6-2
Native/Extended 1-3
Negate Accumulator 5-108
Negate HL instruction 5-10
Negate HL Register (Word) 5-109
No Operation 5-110
No Operation instruction 5-16
NONMASKABLE INTERRUPT 6-5
Nonmaskable Interrupt (NMI) 6-1

O

Object-code compatibility 1-1
ON-CHIP I/O ADDRESS SPACE 2-6
On-Chip I/O Address Space 2-1
On-Chip Register Files 1-1
Opcode Trap 1-1
Operand 4-1
OR (Byte) 5-111
OR (Word) 5-112
Output (Byte) 5-121
Output (to Page 0) 5-124
Output (Word) 5-122
Output Accumulator 5-123
Output and Decrement (Byte) 5-127
Output and Decrement (Word) 5-128
Output and Increment (Byte) 5-129
Output and Increment (Word) 5-130
Output Decrement Memory 5-113
Output Direct to Port Address (Byte) 5-125
Output Direct to Port Address (Word) 5-126
Output Increment Memory 5-117
Output, Decrement and Repeat (Byte) 5-115
Output, Decrement and Repeat (Word) 5-116
Output, Decrement Memory Repeat 5-114
Output, Increment and Repeat (Byte) 5-119
Output, Increment and Repeat (Word) 5-120
Output, Increment Memory Repeat 5-118

P

Parity/Overflow Flag 5-2
Pascal FOR loop 5-12
Pop Accumulator 5-131
Pop Control Register 5-132
Pop operation 2-4
Pop Register 5-133
POPed 2-4
Primary and Working registers 2-1
PROCESSOR FLAGS 5-1
Program Control Group 5-12
Program Counter 2-3
Program Counter Relative 1-4
Program Counter Relative Address 4-1
Program Counter, Interrupt register, Refresh 2-1
Pseudo Pascal 5-17
Push Accumulator 5-134
Push and Pop operations 3-2
Push Control Register 5-135
Push Immediate 5-136
Push Register 5-137
PUSH/POP instructions 5-7
PUSHed 2-4

R

Register (R, RX) 4-1
Register 1-4
Register file 1-1
Register pairs 1-1
Register, Immediate 4-1
Relative Addressing mode 4-6
RESC LCK 5-5
RESC LW (Reset Long Word) 3-3
RESC LW 1-3
RESC LW instruction 5-5
Reserved on-chip I/O instructions 2-6
Reset Bit 5-138
Reset Control Bit 5-139
Reset Control Long Word 1-3
Restart 5-157
Restart instruction 6-5
RETI INSTRUCTION 6-6
Return 5-140
Return From Nonmaskable Interrupt 6-3
Return from Interrupt 5-141
Return from Interrupt 6-6
Return from Nonmaskable Interrupt 5-142
Rotate and Shift Group 5-11
Rotate Digit 4-10
Rotate instructions 5-2
Rotate Left (Accumulator) 5-145
Rotate Left (Byte) 5-143
Rotate Left (Word) 5-144
Rotate Left Circular (Accumulator) 5-148
Rotate Left Circular (Byte) 5-146
Rotate Left Circular (Word) 5-147
Rotate Left Circular A-1
Rotate Left Digit 5-149
Rotate Right (Accumulator) 5-152
Rotate Right (Byte) 5-150
Rotate Right (Word) 5-151
Rotate Right Circular (Accumulator) 5-155
Rotate Right Circular (Byte) 5-153
Rotate Right Circular (Word) 5-154
Rotate Right Digit 5-156
Save/restore 5-7
Saved carry 5-2
Select Register (SR) 2-1
Select Register 5-4, 6-1
Set Bit 5-162
Set Bit and Reset Bit 5-11
Set Carry flag 5-161
Set Control Bit 5-163
Set Control Long Word 1-3
Set/Reset Condition flag 5-12
SETC LCK 5-5
SETC LW (Set Long Word) 3-3
SETC LW 1-3
SETC LW instruction 5-5
SETC XM instruction 5-4
Shift Left Arithmetic (Byte) 5-164
Shift Left Arithmetic (Word) 5-165
Shift Right Arithmetic (Byte) 5-167
Shift Right Arithmetic (Word) 5-168
Shift Right Logical (Byte) 5-169
Shift Right Logical (Word) 5-170
Sign flag 5-2
Sleep 5-166
Source/destination combination 5-7
Stack Pointer (SP) 2-1
Stack Pointer Relative 1-4, 1-5, 4-1
Stack Pointer Relative addressing mode 4-7
Standby Mode 1-1
String search 5-8
Subroutine Call 5-1
Subroutine Return 5-1
Subtract (Byte) 5-171
Subtract (Word) 5-172, 5-174
Subtract from Stack Pointer (Word) 5-173
Subtract With Carry (Word) 5-159
Subtract with Carry (Byte) 5-158
SWAP 1-4
Swap instruction 5-7
Swap Upper Register Word with Lower Register 5-175

T

Test (Byte) 5-176
Test I/O Port 5-177
TRAP INTERRUPT 6-4
Trap and Break Register 2-6, 6-4
Trap handling routine 5-5
Trap on Instruction Fetch 6-4
Trap on Interrupt Vector 6-4
Trap Register 5-15, 6-4
TST instruction 5-9
TSTIO instruction 5-15

U

Ump relative/Call relative 1-6
Unsigned divide instruction 5-10

V

Vectored interrupt mode (Mode 2) 1-5

W

W decoder directive 3-2
Word or Long Word block transfer 5-8
Word strings 4-10
Word/Long Word 1-3

X

Zero byte input 5-2
Zero flag 5-2

Appendix A A

Appendix B B

Appendix C C

Appendix D D

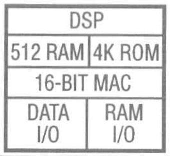
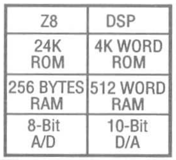
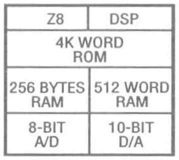
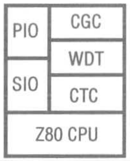
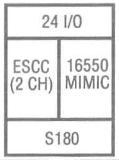
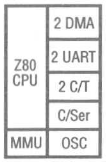
Appendix E E

Index I

Superintegration™ Products Guide **S**

Zilog's Literature Guide Ordering Information L

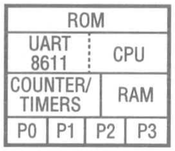
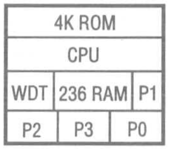
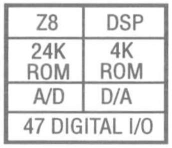
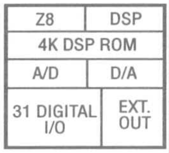
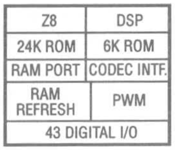
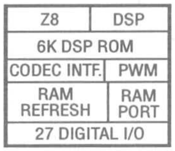
ZiLOG Fax/Modem Superintegration™ Products Guide

	Data Pump	Single Chip		Controllers			
Block Diagram	 DSP 512 RAM 4K ROM 16-BIT MAC DATA I/O RAM I/O	 Z8 DSP 24K ROM 4K WORD ROM 256 BYTES RAM 512 WORD RAM 8-Bit A/D 10-Bit D/A	 Z8 DSP 4K WORD ROM 256 BYTES RAM 512 WORD RAM 8-BIT A/D 10-BIT D/A	 PIO CGC WDT SIO CTC Z80 CPU	 24 I/O ESCC (2 CH) 16550 MIMIC S180	 Z80 CPU 2 DMA 2 UART 2 C/T C/Ser MMU OSC	
Part #	**Z89C00**	**Z89120**	**Z89920**	**Z84C15**	**Z80182**	**Z80180**	**Z85230**
Description	16-Bit Digital Signal Processor	Zilog Modem/Fax Controller (ZMFC)	Zilog Modem/Fax Controller (ZMFC)	IPC/EIPC Controller	Zilog Intelligent Peripheral (ZIP™)	High-performance Z80® CPU with peripherals	Enhanced Serial Com. Controller
Process/Speed	CMOS 10, 15 MHz	CMOS 20 MHz	CMOS 20 MHz	CMOS 6, 10,16 MHz	CMOS 16, 20 MHz	6, 8, 10, 16*, 20* *Z8S180 only	CMOS 8, 10,16, 20 MHz
Features	16-bit Mac 75 ns 2 data RAMs (256 words each) 4K word ROM 64Kx16 Ext. ROM 16-bit I/O Port 74 instructions Most single cycle Two conditional branch inputs, two user outputs Library of software macros available zero overhead pointers	Z8® controller with 24 Kbyte ROM 16-bit DSP with 4K word ROM 8-bit A/D 10-bit D/A (PWM) Library of software macros available 47 I/O pins Two comparators Independent Z8® and DSP Operations Power-Down Mode	Z8 w/64K external memory DSP w/4K word ROM 8-bit A/D 10-bit D/A Library of macros 47 I/O pins Two comparators Independent Z8® and DSP Operations Power-Down Mode	Z80® CPU, SIO, CTC WDT, CGC The Z80 Family in one device Power-On Reset Two chip selects 32-bit CRC WSG EV mode[1] 3 and 5 Volt Version	Complete Static Version of Z180™ plus ESCC (2 channels of Z85230) 16550 MIMIC 24 Parallel I/O Emulation Modes[1]	Enhanced Z80® CPU MMU 1 Mbyte 2 DMAs 2 UARTs with BRGs C/Serial I/O Port Oscillator Z8S180 includes; Pwr dwn, Prgmble EMI, divide-by-one clock option	Full dual-channel SCC plus deeper FIFOs: 4 bytes on Tx 8 bytes on Rx DPLL counter per channel Software compatible to SCC
Package	68-pin PLCC 60-pin VQFP	68-pin PLCC	68-pin PLCC	100-pin QFP 100-pin VQFP	100-pin QFP 100-pin VQFP	64-pin DIP 68-pin PLCC 80-pin QFP	40-pin DIP 44-pin PLCC
Other Applications	16-bit General-Purpose DSP TMS 32010/20/25 applications	Multimedia-Audio Voicemail Speech Storage and Transmission Modems FAXes, Sonabouys	Multimedia-Audio Voicemail Speech Storage and Transmission Modems FAXes, Sonabouys	Intelligent peripheral controllers Modems	General-Purpose Embedded Control Modem, Fax, Data Communications	Embedded Control	General-Purpose datacom. High performance SCC software compatible upgrade

ZiLOG Mass Storage Superintegration™ Products Guide

Block Diagram	UART CPU | OSC 256 RAM | CLOCK P0 | P1 | P2 | P3	8K PROM | UART CPU 256 RAM P0 | P1 | P2 | P3	DSP 512 RAM | 4K ROM 16-BIT MAC DATA I/O | RAM I/O	MULT | DIV | UART CPU | OSC 256 RAM | CLOCK P0 | P1 | P2 | P3	MULT | DIV | UART CPU | DSP DAC | PWM ADC | SPI P2 | P3 | A15-0	88-BIT R-S ECC | SRAM/DRAM CTRL DISK INTER-FACE | MCU INTER-FACE | AT/DE HOST INTER-FACE
Part #	**Z86C91/Z8691**	**Z86E21**	**Z89C00**	**Z86C93**	**Z86C95**	**Z86018**
Description	ROMless Z8®	Z8® 8K OTP	16-Bit Digital Signal Processor	Enhanced Z8®	Enhanced Z8® with DSP	Zilog Datapath Controller (ZDPC)
Process/Speed	CMOS 16 MHz (C91) NMOS 12 MHz (91)	CMOS 12, 16 MHz	CMOS 10, 15 MHz	CMOS 20, 25 MHz	CMOS 24 MHz	CMOS 40 MHz
Features	Full duplex UART 2 Standby Modes (STOP and HALT) 2x8 bit Counter/Timer	8K OTP ROM 256 Byte RAM Full-duplex UART 2 Standby Modes (STOP and HALT) 2 Counter/Timers ROM Protect option RAM Protect option Low EMI option	16-bit Mac 75 ns 2 data RAMs (256 words each) 4K word ROM 64Kx16 Ext. ROM 16-bit I/O Port 74 instructions Most single cycle Two conditional branch inputs, two user outputs Library of software macros available zero overhead pointers	16x16 Multiply 1.7 µs 32x16 Divide 2.0 µs Full duplex UART 2 Standby Modes (STOP and HALT) 3 16-bit Counter/Timers Pin compatible to Z86C91 (PDIP)	8 channel 8-bit ADC, 8-bit DAC 16-bit Multiply/Divide Full duplex UART SPI (Serial Peripheral Interface) 3 Standby Modes (STOP/HALT/PAUSE) Pulse Width Modulator 3x16-bit timer 16-bit DSP slave processor 83 ns Mult./Accum.	Full track read Automatic data transfer (Point & Go®) 88-bit Reed Solomon ECC "on the fly" Full AT/IDE bus interface 64 KB SRAM buffer 1 MB DRAM buffer Split data field support 100-pin VQFP package JTAG boundary scan option Up to 8 KB buffer RAM reserved for MCU
Package	40-pin DIP 44-pin PLCC 44-pin QFP	40-pin DIP 44-pin PLCC 44-pin QFP	68-pin PLCC 60-pin VQFP	40-pin DIP 44-pin PLCC 44-pin QFP 48-pin VQFP	80-pin QFP 84-pin PLCC 100-pin VQFP	100-pin VQFP 100-pin QFP
Application	Disk Drives Modems Tape Drives	Software Debug Z8® prototyping Z8® production runs Card Reader	Disk Drives Tape Drives Servo Control Motor Control	Disk Drives Tape Drives Modems	Disk Drives Tape Drives Servo Control Motor Control	Hard Disk Drives

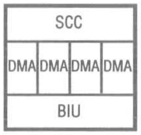
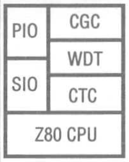
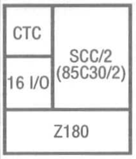
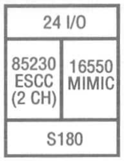
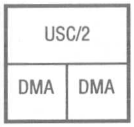
Zilog Telephone Answering Devices Superintegration™ Products Guide

Block Diagram	 ROM UART 8611 CPU COUNTER/ TIMERS RAM P0 P1 P2 P3	 4K ROM CPU WDT 236 RAM P1 P2 P3 P0	 Z8 DSP 24K ROM 4K ROM A/D D/A 47 DIGITAL I/O	 Z8 DSP 4K DSP ROM A/D D/A 31 DIGITAL I/O EXT. OUT	 Z8 DSP 24K ROM 6K ROM RAM PORT CODEC INTF. RAM REFRESH PWM 43 DIGITAL I/O	 Z8 DSP 6K DSP ROM CODEC INTF. PWM RAM REFRESH RAM PORT 27 DIGITAL I/O
Part #	**Z08600/Z08611**	**Z86C30/E30** **Z86C40/E40**	**Z89C65**	**Z89C66**	**Z89C67**	**Z89C68**
Description	Z8® NMOS (CCP™) 8600 = 2K ROM 8611 = 4K ROM	Z8® Consumer Controller Processor (CCP™) with 4K ROM C30 = 28-pin C40 = 40-pin E30/E40 = OTP version	Telephone Answering Controller with DSP LPC voice synthesis and DTMF detection	Telephone Answering Controller with DSP LPC voice synthesis and DTMF detection and external ROM/RAM interface	Telephone Answering Controller with digital voice encode and decode DTMF detection and full memory control interface	Telephone Answering Controller with digital voice encode and decode DTMF detection and external ROM/RAM interface
Process/Speed	NMOS 8,12 MHz	CMOS 12 MHz	CMOS 20 MHz	CMOS 20 MHz	CMOS 20 MHz	CMOS 20 MHz
Features	2K/4K ROM 128 Bytes RAM 22/32 I/O lines On-chip oscillator 2 Counter/Timers 6 vectored, priority interrupts UART (Z8611)	4K ROM, 236 RAM 2 Standby Modes 2 Counter/Timers ROM Protect RAM Protect 4 Ports (86C40/E40) 3 Ports (86C30/E30) Brown-Out Protection 2 Analog Comparators Low EMI Watch-Dog Timer Auto Power-On Reset Low Power option	Z8® Controller 24K ROM 16-bit DSP 4K Word ROM 8-bit A/D with AGC DTMF macro available LPC macro available 10-bit PWM D/A Other DSP software options available 47 I/O Pins	Z8® Controller 16-bit DSP 4K Word ROM 8-bit A/D with AGC DTMF macro available LPC macro available 10-bit PWM D/A Other DSP software options available External ROM/RAM capability 31 I/O Pins	Z8® Controller 24K ROM 16-bit DSP 6K Word ROM DTMF macro available LPC macro available 10-bit PWM D/A Other DSP software options available ARAM/DRAM/ROM Controller & Interface Dual Codec Interface 43 I/O	Z8® Controller 64K ROM (external) 16-bit DSP, 6K word ROM DTMF macro available LPC macro available 10-bit PWM D/A Other DSP software options available ARAM/DRAM control/ interface External ROM/RAM Dual Codec Interface 27 I/O
Package	28-pin DIP 40-pin DIP 44-pin PLCC	28-pin DIP 40-pin DIP 44-pin PLCC, QFP	68-pin PLCC	68-pin PLCC	84-pin PLCC	84-pin PLCC
Application	Low cost tape board TAD	Window Control Wiper Control Sunroof Control Security Systems TAD	Fully featured cassette answering machines with voice prompts and DTMF signaling	General-Purpose DSP applications in TAD and other high-performance 1-tape voice processors	Voice Processing, DSP applications in tapeless TAD and other high-performance 1-tape voice processors	Voice Processing, DSP applications in tapeless TAD and other high-performance 1-tape voice processors

ZiLOG Video Products Superintegration™ Products Guide

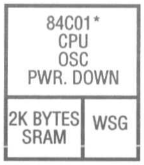
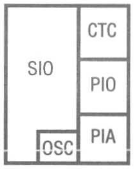
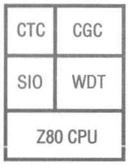
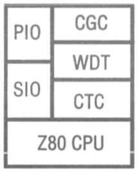
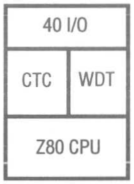
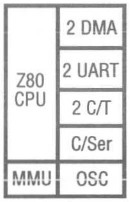
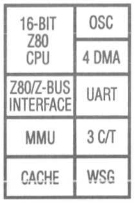
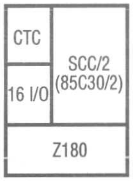
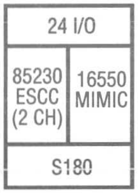
	TV Controller			IR Controller		Cable TV	
Block Diagram	8K ROM 4K CHAR ROM Z8 CPU | RAM OSD 13 PWM | TIMER WDT | 5 PORTS	6K ROM 3K CHAR ROM Z8 CPU | RAM OSD 7 PWM | TIMER WDT | 3 PORTS	CHAR ROM COMMAND INTERPRETER ANALOG SYNC/DATA SLICER | OSD CTRL	1K/6K ROM Z8 CPU WDT | 124 RAM P2 | P3	2K/8K/16K ROM Z8 CPU WDT | 128,256, 768 RAM P0 | P1 | P2 | P3	4K ROM CPU WDT | 236 RAM | P1 P2 | P3 | P0	16K ROM | UART CPU | 236 RAM P0 | P1 | P2 P3 | P4 | P5 | P6
Part #	**Z86C27/127/97**	**Z86227**	**Z86128**	**Z86L06/L29**	**Z86L70/71/72** (Q193)	**Z86C40/E40**	**Z86C61/62**
Description	Z8® Digital Television Controller MCU with logic functions needed for Television Controller, VCRs and Cable	Standard DTC features with reduced ROM, RAM, PWM outputs for greater economy	Line 21 Controller (L21C™) for Closed Caption Television	18-pin Z8® Consumer Controller Processor (CCP™) low-voltage and low-current battery operation 1K-6K ROM	Z8® (CCP™) low-voltage parts that have more ROM, RAM and special Counter/Timers for automated output drive capabilities	Z8® Consumer Controller Processor (CCP™) with 4K ROM (C40) E40 = OTP version	Z8® MCU with Expanded I/O's and 16K ROM
Process/Speed	CMOS 4 MHz	CMOS 4 MHz	CMOS 12 MHz	Low Voltage CMOS 8 MHz	Low Voltage CMOS 8 MHz	CMOS 12 MHz	CMOS 16, 20 MHz
Features	Z8/DTC Architecture 8K ROM, 256-byte RAM 160x7-bit video RAM On-Screen Display (OSD) video controller Programmable color, size, position attributes 13 PWMs for D/A conversion 128-character set 4Kx6-bit char. Gen. ROM Watch-Dog Timer (WDT) Brown-Out Protection 5 Ports/36 pins 2 Standby Modes Low EMI Mode	Z8/DTC Architecture 6K ROM, 256-byte RAM 120x7-bit video RAM OSD on board Programmable color, size, position attributes 7 PWMs 96-character set 3Kx6-bit character generator ROM Watch-Dog Timer (WDT) Brown-Out Protection 3 Ports/20 pins 2 Standby Modes Low EMI Mode	Conforms to FCC Line 21 format Parallel or serial modes Stand-alone operation On-board data sync and slicer On-board character generator - Color - Blinking - Italic - Underline	Z8® Architecture 1K ROM & 6K ROM Watch-Dog Timer 2 Analog Comparators with output option 2 Standby Modes 2 Counter/Timers Auto Power-On Reset 2 volt operation RC OSC option Low Noise option Brown-Out Protection High current drivers (2, 4)	Z8® Architecture 2K/8K/16K ROM Watch-Dog Timer 2 Analog Comparators with output option 2 Standby Modes 2 Enhanced Counter/ Timers, Auto Pulse Reception/Generation Auto Power-On Reset 2 volt operation RC OSC option Brown-Out Protection High current drivers (4)	4K ROM, 236 RAM 2 Standby Modes 2 Counter/Timers ROM Protect RAM Protect 4 Ports Brown-Out Protection 2 Analog Comparators Low EMI Watch-Dog Timer Auto Power-On Reset Low Power option	16K ROM Full duplex UART 2 Standby Modes (STOP and HALT) 2 Counter/Timers ROM Protect option RAM Protect option Pin compatible to Z86C21 C61 = 4 Ports C62 = 7 Ports
Package	64-pin DIP 52-pin active (127)	40-pin DIP	18-pin DIP	18-pin DIP 18-pin SOIC	20-pin DIP (L71), 18-pin DIP, SOIC (L70) 40,44-pin DIP, PLCC, QFP (L72)	40-pin DIP	40-pin DIP (C61) 44-pin PLCC,QFP (C61) 68-pin PLCC (C62)
Application	Low-end Television Cable/Satellite Receiver	Low-end Television Cable/Satellite Receiver	TVs, VCRs, Decoders	I.R. Controller Portable battery operations	I.R. Controller Portable battery operations	Window Control Wiper Control Sunroof Control Security Systems TAD	Cable Television Remote Control Security

ZiLOG Datacommunications Superintegration™ Products Guide

Block Diagram									
Part #	**Z8030/Z80C30 Z8530/Z85C30**	**Z85230/Z80230 Z85233***	**Z16C35**	**Z84C15**	**Z80181**	**Z80182**	**Z16C30**	**Z16C33**	**Z16C32**
Description	Serial Com. Controller	Enhanced Serial Com. Controller	Integrated Serial Com. Controller	Intelligent Peripheral Controller	Smart Access Controller	Zilog Intelligent Peripheral	Universal Serial Controller	Mono-channel Universal Serial Controller	Integrated Universal Serial Controller
Process/ Speed/ Clock Data Rate	NMOS: 4, 6, 8 MHZ CMOS: 8,10 16 MHz 2, 2.5, 4 Mb/s	CMOS: 10, 16 20 MHz 2.5, 4.0, 5.0 Mb/s	CMOS: 10, 16 MHz 2.5, 4.0 Mb/s	CMOS 6, 10,16 MHz	10, 12.5	CMOS 16, 20 MHz	CMOS: 20 MHz CPU Bus 10 Mb/s 20 Mb/s	CMOS: 10 MHz CPU Bus 10 Mb/s	CMOS:20 MHz CPU Bus 16 Mb/s 20 Mb/s
Features	Two independent full-duplex channels Enhanced DMA support: 10x19 status FIFO 14-bit byte counter NRZ/NRZI/FM	Full dual-channel SCC plus deeper FIFOs: 4 bytes on Tx 8 bytes on Rx DPLL counter per channel Software compatible to SCC *One channel of Z85230	Full dual-channel SCC plus 4 DMA controllers and a bus interface unit	Z80® CPU, SIO, CTC WDT, CGC The Z80 Family in one device Power-On Reset Two chip selects 32-bit CRC WSG EV mode[1] 3 and 5 Volt Version	Complete Z180™ plus SCC/2 CTC 16 I/O lines Emulation Mode[1]	Complete Static version of Z180 plus ESCC (2 channels of 85230) 16550 MIMIC 24 Parallel I/O Emulation Mode[1]	Two dual-channel 32-byte receive & transmit FIFOs 16-bit bus B/W: 18.2 Mb/s 2 BRGs per channel Flexible 8/16-bit bus interface	Single-channel (half of USC™) plus Time Slot Assigner functions for ISDN	Single-channel (half of USC) plus two DMA controllers Array chained and linked-list modes with ring buffer support
Package	40-pin DIP 44-pin CERDIP 44-pin PLCC	40-pin DIP 44-pin PLCC *44-pin QFP (85233)	68-pin PLCC	100-pin QFP 100-pin VQFP	100-pin QFP	100-pin QFP 100-pin VQFP	68-pin PLCC	68-pin PLCC	68-pin PLCC
Application	General-Purpose datacom.	General-Purpose datacom. High performance SCC software	High performance datacom. SCC upgrades	Intelligent peripheral controllers Modems	Intelligent peripheral controllers Printers, Faxes, Modems, Terminals	General-Purpose Embedded Control Modem, Fax, Data Communications	General-Purpose high-end datacom. Ethernet HDLC X.25 Frame Relay	General-Purpose high-end datacom. Ethernet HDLC X.25 Frame Relay	General-Purpose high-end datacom. Ethernet HDLC X.25 Frame Relay

AppleTalk® A Registered Trademark of Apple Computer, Inc.

ZiLOG Z80® Embedded Controllers Superintegration™ Products Guide

Block Diagram	 84C01* CPU OSC PWR. DOWN 2K BYTES SRAM WSG	 SIO CTC PIO OSC PIA	 CTC CGC SIO WDT Z80 CPU	 PIO CGC WDT SIO CTC Z80 CPU	 40 I/O CTC WDT Z80 CPU	 Z80 CPU 2 DMA 2 UART 2 C/T C/Ser MMU OSC	 16-BIT Z80 CPU OSC 4 DMA Z80/Z-BUS INTERFACE UART MMU 3 C/T CACHE WSG	 CTC 16 I/O SCC/2 (85C30/2) Z180	 24 I/O 85230 ESCC (2 CH) 16550 MIMIC S180
Part #	**Z84C50**	**Z84C90**	**Z84013/C13**	**Z84015/C15**	**Z84011/C11**	**Z80180/S180**	**Z80280**	**Z80181**	**Z80182**
Description	Z80/84C01 with 2K SRAM	Killer I/O (3 Z80 peripherals)	Intelligent Peripheral Controller	Intelligent Peripheral Controller	Parallel I/O Controller	High-performance Z80® CPU with peripherals	16-bit Z80® code compatible CPU with peripherals	Smart Access Controller	Zilog Intelligent Peripheral
Speed MHz	10	8, 10, 12.5	6, 10	6, 10, 16	6, 10	6, 8, 10, 16*, 20* *Z8S180 only	10, 12	10, 12.5	16, 20
Features	Z80® CPU 2 Kbytes SRAM WSG Oscillator Pin compatible with Z84C00 DIP & PLCC EV mode[1] *84C01 is available as a separate part	SIO, PIO, CTC plus 8 I/O lines	Z80® CPU, SIO, CTC WDT, CGC, WSG, Power-On Reset 2 chip selects EV mode[1]	Z80® CPU, SIO, CTC WDT, CGC The Z80 Family in one device Power-On Reset Two chip selects 32-bit CRC WSG EV mode[1]	Z80® CPU, CTC, WDT 40 I/O lines bit programmable Power-On Reset EV mode[1]	Enhanced Z80 CPU MMU 1 Mbyte 2 DMAs 2 UARTs with BRGs C/Serial I/O Port Oscillator Z8S180 includes; Pwr dwn, Prgmble EMI, divide-by-one clock option	16-bit code compatible Z80® CPU Three stage pipeline MMU 16 Mbyte CACHE 256 byte Inst. & Data Peripherals 4 DMAs, UART, 3 16-bit C/T, WSG Z80/Z-BUS® interface	Complete Z180 plus SCC/2 CTC 16 I/O lines Emulation Mode[1]	Complete Static Version of Z180™ plus ESCC (2 channels of Z85230) 16550 MIMIC 24 Parallel I/O Emulation Modes[1]
Package	40-pin DIP 44-pin PLCC 44-pin QFP	84-pin PLCC	84-pin PLCC	100-pin QFP 100-pin VQFP	100-pin QFP	64-pin DIP 68-pin PLCC 80-pin QFP	68-pin PLCC	100-pin QFP	100-pin QFP 100-pin VQFP
Application	Embedded Controllers	General-purpose peripheral that can be used with Z80 and other CPU's	Intelligent datacom controllers	Intelligent peripheral controllers Modems	Intelligent parallel-I/O controllers Industrial display terminals	Embedded Control	Embedded Control Terminals Printers	Intelligent peripheral controllers Printers, Faxes, Modems, Terminals	General-Purpose Embedded Control Modem, Fax, Data Communications

---

[1] Allows use of existing development systems.

ZiLOG Peripherals Superintegration™ Products Guide

	Z8036 Z8536	Z32H00	Z5380 Z53C80	Z85C80
Description	Counter/Timer & parallel I/O Unit (CIO)	Hyperstone Enhanced Fast Instruction Set Computer (EFISC) Embedded (RISC) Processor	Small Computer System Interface (SCSI)	Serial Communication Controller and Small Computer System Interface
Process/ Speed	NMOS 4,6 MHz	CMOS 25 MHz	CMOS Z5380: 1.5 MB/s Z53C80: 3.0 MB/s	CMOS SCC - 10, 16 MHz SCSI - 3.0 MB/s
Features	Three 16-bit Counter/Timers, Three I/O ports with bit catching, pattern matching interrupts and handshake I/O	32-bit MPU 4 Gbytes address space 19 global and 64 local registers of 32 bits each 128 bytes instruction cache 1.2µ CMOS 42 mm² die	ANSI X3.131-1986 Direct SCSI bus interface On-board 48 mA drivers Normal or Block mode DMA transfers Bus interface, target and initiator	Full dual-channel SCC plus SCSI sharing databus and read/write functions
Package	40-pin PDIP 44-pin PLCC	144-pin PGA 132-pin QFP	Z5380: 40-pin DIP 44-pin PLCC Z53C80: 48-pin DIP 44-pin PLCC	68-pin PLCC
Application	General-Purpose Counter/Timers and I/O system designs	Embedded high-performance industrial controller Workstations	Bus host adapters, formatters, host ports	AppleTalk® networking SCSI disk drives

[2] Software and hardware compatible with discrete devices.

ZiLOG

Appendix A	A
Appendix B	B
Appendix C	C
Appendix D	D
Appendix E	E
Index	I
Superintegration™ Products Guide	S
Zilog's Literature Guide Ordering Information	**L**

ZiLOG

LITERATURE GUIDE

Z8®/SUPER8™ MICROCONTROLLER FAMILY

Databooks	Part No	Unit Cost
Z8 Microcontrollers Databook (includes the following documents)	**DC-8275-04**	**5.00**

Z8 CMOS Microcontrollers

Z86C00/C10/C20 MCU OTP Product Specification
Z86C06 Z8 CCP™ Preliminary Product Specification
Z86C08 8-Bit MCU Product Specification
Z86E08 Z8 OTP MCU Product Specification
Z86C09/19 Z8 CCP Product Specification
Z86E19 Z8 OTP MCU Advance Information Specification
Z86C11 Z8 MCU Product Specification
Z86C12 Z8 ICE Product Specification
Z86C21 Z8 MCU Product Specification
Z86E21/Z86E22 OTP Product Specification
Z86C30 Z8 CCP Product Specification
Z86E30 Z8 OTP CCP Product Specification
Z86C40 Z8 CCP Product Specification
Z86E40 Z8 OTP CCP Product Specification
Z86C27/97 Z8 DTC™ Product Specification
Z86127 Low-Cost Digital Television Controller Adv. Info. Spec.
Z86C50 Z8 CCP ICE Advance Information Specification
Z86C61 Z8 MCU Advance Information Specification
Z86C62 Z8 MCU Advance Information Specification
Z86C89/C90 CMOS Z8 CCP Product Specification
Z86C91 Z8 ROMless MCU Product Specification
Z86C93 Z8 ROMless MCU Preliminary Product Specification
Z86C94 Z8 ROMless MCU Product Specification
Z86C96 Z8 ROMless MCU Advance Information Specification
Z88C00 CMOS Super8 MCU Advance Information Specification

Z8 NMOS Microcontrollers

Z8600 Z8 MCU Product Specification
Z8601/03/11/13 Z8 MCU Product Specification
Z8602 8-Bit Keyboard Controller Preliminary Product Spec.
Z8604 8-Bit MCU Product Specification
Z8612 Z8 ICE Product Specification
Z8671 Z8 MCU With BASIC/Debug Interpreter Product Spec.
Z8681/82 Z8 MCU ROMless Product Specification
Z8691 Z8 MCU ROMless Product Specification
Z8800/01/20/22 Super8 ROMless/ROM Product Specification

Peripheral Products

Z86128 Closed-Captioned Controller Adv. Info. Specification
Z765A Floppy Disk Controller Product Specification
Z5380 SCSI Product Specification
Z53C80 SCSI Advance Information Specification

Z8 Application Notes and Technical Articles

Zilog Family On-Chip Oscillator Design
Z86E21 Z8 Low Cost Thermal Printer
Z8 Applications for I/O Port Expansions
Z86C09/19 Low Cost Z8 MCU Emulator
Z8602 Controls A 101/102 PC/Keyboard
The Z8 MCU Dual Analog Comparator
The Z8 MCU In Telephone Answering Systems
Z8 Subroutine Library
A Comparison of MCU Units
Z86xx Interrupt Request Registers
Z8 Family Framing
A Programmer's Guide to the Z8 MCU
Memory Space and Register Organization

Super8 Application Notes and Technical Articles

Getting Started with the Zilog Super8
Polled Async Serial Operations with the Super8
Using the Super8 Interrupt Driven Communications
Using the Super8 Serial Port with DMA
Generating Sine Waves with Super8
Generating DTMF Tones with Super8
A Simple Serial Parallel Converter Using the Super8

Additional Information

Z8 Support Products
Zilog Quality and Reliability Report
Literature List
Package Information
Ordering Information

Z8®/SUPER8™ MICROCONTROLLER FAMILY (Continued)

Databooks	Part No	Unit Cost
Digital Signal Processor Databook (includes the following documents) Z86C95 Z8® Digital Signal Processor Preliminary Product Specification Z89C00 16-Bit Digital Signal Processor Preliminary Product Specification Z89C00 DSP Application Note "Understanding Q15 Two's Complement Fractional Multiplication" Z89120, Z89920 (ROMless) 16-Bit Mixed Signal Processor Preliminary Product Specification Z89121, Z89921 (ROMless) 16-Bit Mixed Signal Processor Preliminary Product Specification Z89320 16-Bit Digital Signal Processor Preliminary Product Specification Z89321 16-Bit Digital Signal Processor Advance Information Specification	DC-8299-02	3.00
Telephone Answering Device Databook (includes the following documents) Z89C65, Z89C66 (ROMless) Dual Processor T.A.M. Controller Preliminary Product Specification Z89C67, Z89C68 (ROMless) Dual Processor Tapeless T.A.M. Controller Preliminary Product Specification	DC-8300-01	3.00
Infrared Remote (IR) Control Databook (includes the following documents) Z86L06 Low Voltage CMOS Consumer Controller Processor Preliminary Product Specification Z86L29 6K Infrared (IR) Remote (ZIRC™) Controller Advance Information Specification Z86L70/L71/L72, Z86E72 Zilog IR (ZIRC™) CCP™ Controller Family Preliminary Product Specification	DC-8301-01	3.00

Z8 Product Specifications, Technical Manuals and Users Guides	Part No	Unit Cost
Z86E23 CMOS Z8 OTP Microcontroller Preliminary Product Specification	DC-2598-00	N/C
Z86C27/97 Z8 DTC™ Product Specification and Addendum	DC-2561-01	N/C
Z86127 Low-Cost Digital Television Controller Preliminary Product Specification	DC-2574-00	N/C
Z86227 40-Pin Low-Cost Digital Television Controller Preliminary Product Specification and Addendum	DC-3002-00	N/C
Z86C61/62/96 CMOS Z8 Microcontroller Preliminary Product Specification	DC-2587-00	N/C
Z86C93 CMOS Z8 ROMless Microcontroller Product Specification	DC-2508-03	N/C
Z88C00 CMOS Super8 ROMless Microcontroller Preliminary Product Specification	DC-2551-00	N/C
Z8614 NMOS Z8 8-Bit MCU Keyboard Controller Preliminary Product Specification	DC-2576-00	N/C
Z86128 Closed-Captioned Controller Preliminary Product Specification and Addendum	DC-2570-01	N/C
Z86017 PCMCIA Adaptor Chip Advance Information Specification	DC-2643-0A	N/C
Z8 OTP CMOS One-Time-Programmable Microcontrollers Addendum	DC-2614-AA	N/C
asm S8 Super8/Z8 Cross Assembler User's Guide	DC-8267-05	3.00
Z8 Microcontrollers Technical Manual	DC-8291-02	5.00
Z86018 Preliminary User's Manual	DC-8296-00	N/C
Digital TV Controller User's Manual	DC-8284-01	3.00
Z89C00 16-Bit Digital Signal Processor User's Manual	DC-8294-01	3.00
PLC Z89C00 Cross Development Tools Brochure	DC-5538-01	N/C

Z8 Application Notes	Part No	Unit Cost
Z8602 Controls A 101/102 PC/Keyboard	DC-2601-01	N/C
The Z8 MCU Dual Analog Comparator	DC-2516-01	N/C
Z8 Applications for I/O Port Expansions	DC-2539-01	N/C
Z86E21 Z8 Low Cost Thermal Printer	DC-2541-01	N/C
Zilog Family On-Chip Oscillator Design	DC-2496-01	N/C
Using the Zilog Z86C06 SPI Bus	DC-2584-01	N/C
Interfacing LCDs to the Z8	DC-2592-01	N/C
X-10 Compatible Infrared (IR) Remote Control	DC-2591-01	N/C
Z86C17 In-Mouse Applications	DC-3001-01	N/C
Z86C40/E40 MCU Applications Evaluation Board	DC-2604-01	N/C
Z86C08/C17 Controls A Scrolling LED Message Display	DC-2605-01	N/C
Z86C95 Hard Disk Controller Flash EPROM Interface	DC-2639-01	N/C
Timekeeping with Z8; DTMF Tone Generation; Serial Communication Using the CCP Software UART	DC-2645-01	N/C

ZiLOG

LITERATURE GUIDE

Z80®/Z8000® CLASSIC FAMILY OF PRODUCTS

Z80®/Z180™/Z280®/Z8000® and Datacom Family	Part No	Unit Cost
Volume I Databook Microprocessors and Peripherals	DC-2610-01	5.00

Discrete Z80® Family

- Z8400/C00 NMOS/CMOS Z80® CPU Product Specification
- Z8410/C10 NMOS/CMOS Z80 DMA Product Specification
- Z8420/C20 NMOS/CMOS Z80 PIO Product Specification
- Z8430/C30 NMOS/CMOS Z80 CTC Product Specification
- Z8440/Z84C40 NMOS/CMOS Z80 SIO Product Specification

Embedded Controllers

- Z84C01 Z80 CPU with CGC Product Specification
- Z84C50 RAM80™ Preliminary Product Specification
- Z8470 Z80 DART Product Specification
- Z84C90 CMOS Z80 KIO™ Product Specification
- Z84011/C11 PIO Parallel I/O Product Specification
- Z84013/015 Z84C13/C15 IPC/EIPC™ Product Specification
- Z80180/Z8S180 Z180 MPU Product Specification
- Z80181 ZIO™ Controller Product Specification
- Z280™ MPU Preliminary Product Specification

Serial Communications Controllers

- Z8030/Z8530 Z-BUS® SCC Product Specification
- Z80C30/Z85C30 SCC Product Specification
- Z85230 ESCC™ Product Specification
- Z80230 Z-BUS ESCC Product Specification
- Z16C35 ISCC™ Product Specification
- Z5380 SCSI Product Specification
- Z53C80 SCSI Product Specification
- Z85C80 SCSI/SCC Product Specification
- Z16C30 USC™ Product Specification
- Z16C32 IUSC™ Product Specification
- Z16C33 MUSC™ Product Specification
- Z16C50 DDPLL™ Product Specification

Technical Articles

- Z80 Questions and Answers
- Z180 Questions and Answers
- SCC Questions and Answers
- ESCC Questions and Answers
- ISCC Questions and Answers

Additional Information

- Superintegration Products Guide
- Support Product Summary
- Product Support
- Military Qualified Products
- Quality and Reliability
- Literature Guide
- Package Information
- Ordering Information

Z80®/Z8000® CLASSIC FAMILY OF PRODUCTS (Continued)

Z80®/Z180™/Z280®/Z8000® and Datacom Family	Part No	Unit Cost
Volume II Databook Microprocessors and Peripherals	**DC-2622-01**	**5.00**

Application Notes

- Z80® Family Interrupt Structure
- Using the Z80® SIO with SDLC
- Using the Z80® SIO In Asynchronous Communications
- Binary Synchronous Communication Using the Z80® SIO
- Serial Communication with the Z80A DART
- Interfacing 8500 Peripherals to the Z80®
- Serial Clock Generation Using the Z8536 CIO
- Timing in an Interrupt-Based System with the Z80® CTC
- A Z80-Based System Using the DMA with the SIO
- Interfacing the Z8500 Peripherals to the 68000
- Z180™ Break Detection
- The Z180™ Interfaced with the SCC at 10 MHz
- Technical Considerations When Implementing LocalTalk Link Access Protocol
- Using the Z84C11/C13/C15 in Place of the Z84011/013/015
- Using SCC with Z8000 in SDLC Protocol
- SCC in Binary Synchronous Communications
- On-Chip Oscillator Design
- Interfacing Z80® CPUs to the Z8500 Peripheral Family
- Boost Your System Performance Using The Zilog ESCC™
- Design a Serial Board to Handle Multiple Protocols
- Using the Z16C30 USC Universal Serial Controller
- Datacommunications IUSC/MUSC Time Slot Assigner
- Integrating Serial Data and SCSI Peripheral Control on One Chip
- A Fast Z80® Embedded Controller
- Using the Zilog Datacom Family with the 80186 CPU

Questions and Answers

- Z80® Questions and Answers
- Z180™ Questions and Answers
- SCC Questions and Answers
- ESCC Questions and Answers
- ISCC Questions and Answers

Additional Information

- Classic Family
- Datacom Products
- Literature Guide

Z80/Z180/Z280 Product Specifications, Technical Manuals and Users Guides	Part No	Unit Cost
Z80 CPU Central Processing Unit Technical Manual	DC-0029-05	3.00
Z80 Family Programmer's Reference Guide	DC-0012-04	3.00
Z80 DMA Direct Memory Access Technical Manual	DC-2013-A0	3.00
Z80 PIO Parallel Input/Output Technical Manual	DC-0008-03	3.00
Z80 CTC Counter/Timer Circuit Technical Manual	DC-0036-03	3.00
Z80 SIO Serial I/O Technical Manual	DC-3033-01	3.00
Z80180 Z180 MPU Microprocessor Unit Technical Manual	DC-8276-04	3.00
Z280 MPU Microprocessor Unit Technical Manual	DC-8224-03	3.00
Z80180/Z8S180 Z180 Microprocessor Product Specification	DC-2609-03	N/C
Z80182 Zilog Intelligent Peripheral (ZIP™)	DC-2616-03	N/C
Z380 Preliminary Product Specification	DC-3003-02	N/C

Z80/Z180/Z280 Application Notes	Part No	Unit Cost
Z180/SCC™ Serial Communications Controller Interface at 10 MHz	DC-2521-02	N/C
Z80 Using the 84C11/C13/C15 in place of the 84011/013/015	DC-2499-02	N/C
A Fast Z80 Embedded Controller	DC-2578-01	N/C

ZiLOG

LITERATURE GUIDE

Z8000® MICROPROCESSOR FAMILY

Z8000 Product Specifications, Technical Manuals and Users Guides	Part No	Unit Cost
Z8000 CPU Central Processing Unit Technical Manual	DC-2010-06	3.00
SCC Serial Communication Controller User's Manual	DC-8293-02	3.00
Z8036 Z-CIO/Z8536 CIO Counter/Timer and Parallel Input/Output Technical Manual	DC-2091-02	3.00
Z8038 Z8000 Z-FIO FIFO Input/Output Interface Technical Manual	DC-2051-01	3.00
Z8000 CPU Central Processing Unit Programmer's Pocket Guide	DC-0122-03	3.00
Z85233 EMSCC Enhanced Mono Serial Communication Controller Preliminary Product Specification	DC-2590-00	N/C
Z85C80 SCSCI™ Serial Communication and Small Computer Interface Preliminary Product Specification	DC-2534-02	N/C
Z16C30 USC Universal Serial Controller Preliminary Technical Manual	DC-8280-02	3.00
Z16C33 CMOS USC/MUSC™ Universal Serial Controller Technical Manual	DC-8285-02	3.00
Z16C32 IUSC™ Integrated Universal Serial Controller Product Specification	DC-2600-00	N/C
Z16C32 IUSC Integrated Universal Serial Controller Product Specification Addendum	DC-2600-00A	N/C
Z16C32 IUSC Integrated Universal Serial Controller Technical Manual	DC-8292-03	3.00
Z16C35 ISCC Integrated Serial Communication Controller Technical Manual	DC-8286-01	3.00
Z16C35 ISCC Integrated Serial Communication Controller Addendum	DC-8286-01A	N/C
Z53C80 Small Computer System Interface (SCSI) Product Specification	DC-2575-01	N/C
Z80230 Z-BUS® ESCC Enhanced Serial Communication Controller Preliminary Product Specification	DC-2603-01	N/C

Z8000 Application Notes	Part No	Unit Cost
Z16C30 Using the USC in Military Applications	DC-2536-01	N/C
Datacom IUSC/MUSC Time Slot Assigner	DC-2497-02	N/C
Datacom Evaluation Board Using The Zilog Family With The 80186 CPU	DC-2560-03	N/C
Boost Your System Performance Using the Zilog ESCC Controller	DC-2555-02	N/C
Z16C30 USC - Design a Serial Board for Multiple Protocols	DC-2554-01	N/C
Using a SCSI Port for Generalized I/O	DC-2608-01	N/C

MILITARY COMPONENTS FAMILY

Military Specifications	Part No	Unit Cost
Z8681 ROMless Microcomputer Military Product Specification	DC-2392-02	N/C
Z8001/8002 Military Z8000 CPU Central Processing Unit Military Product Specification	DC-2342-03	N/C
Z8581 Military CGC Clock Generator and Controller Military Product Specification	DC-2346-01	N/C
Z8030 Military Z8000 Z-SCC Serial Communications Controller Military Product Specification	DC-2388-02	N/C
Z8530 Military SCC Serial Communications Controller Military Product Specification	DC-2397-02	N/C
Z8036 Military Z8000 Z-CIO Counter/Timer Controller and Parallel I/O Military Electrical Specification	DC-2389-01	N/C
Z8038/8538 Military FIO FIFO Input/Output Interface Unit Military Product Specification	DC-2463-02	N/C
Z8536 Military CIO Counter/Timer Controller and Parallel I/O Military Electrical Specification	DC-2396-01	N/C
Z8400 Military Z80 CPU Central Processing Unit Military Electrical Specification	DC-2351-02	N/C
Z8420 Military PIO Parallel Input/Output Controller Military Product Specification	DC-2384-02	N/C
Z8430 Military CTC Counter/Timer Circuit Military Electrical Specification	DC-2385-01	N/C
Z8440/1/2/4 Z80 SIO Serial Input/Output Controller Military Product Specification	DC-2386-02	N/C
Z80C30/85C30 Military CMOS SCC Serial Communications Controller Military Product Specification	DC-2478-02	N/C
Z84C00 CMOS Z80 CPU Central Processing Unit Military Product Specification	DC-2441-02	N/C
Z84C20 CMOS Z80 PIO Parallel Input/Output Military Product Specification	DC-2384-02	N/C
Z84C30 CMOS Z80 CTC Counter/Timer Circuit Military Product Specification	DC-2481-01	N/C
Z84C40/1/2/4 CMOS Z80 SIO Serial Input/Output Military Product Specification	DC-2482-01	N/C
Z16C30 CMOS USC Universal Serial Controller Military Preliminary Product Specification	DC-2531-01	N/C
Z80180 Z180 MPU Microprocessor Unit Military Product Specification	DC-2538-01	N/C
Z84C90 CMOS KIO Serial/Parallel/Counter Timer Preliminary Military Product Specification	DC-2502-00	N/C
Z85230 ESCC Enhanced Serial Communication Controller Military Product Specification	DC-2595-00	N/C

GENERAL LITERATURE

Catalogs, Handbooks and Users Guides	Part No	Unit Cost
Superintegration Shortform Catalog 1992	DC-5472-11	N/C
Superintegration Products Guide	DC-5499-07	N/C
ZIA™3.3-5.5V Matched Chip Set for AT Hard Disk Drives Datasheet	DC-5556-01	N/C
ZIA ZIA00ZCO Disk Drive Development Kit Datasheet	DC-5593-01	N/C
Zilog Hard Disk Controllers - Z86C93/C95 Datasheet	DC-5560-01	N/C
Zilog Infrared (IR) Controllers - ZIRC™ Datasheet	DC-5558-01	N/C
Zilog Intelligent Peripheral Controller - ZIP™Z80182 Datasheet	DC-5525-01	N/C
Zilog Digital Signal Processing - Z89320 Datasheet	DC-5547-01	N/C
Zilog Datacommunications Brochure	DC-5519-00	N/C
Zilog Digital Signal Processing Brochure	DC-5536-02	N/C
Zilog PCMCIA Adaptor Chip Z86017 Datasheet	DC-5585-01	N/C
Zilog Television/Video Controllers Datasheet	DC-5567-01	N/C
Zilog TAD Controllers - Z89C65/C67/C69 Datasheet	DC-5561-01	N/C
Quality and Reliability Report	DC-2475-11	N/C
The Handling and Storage of Surface Mount Devices User's Guide	DC-5500-02	N/C
Universal Object File Utilities User's Guide	DC-8236-04	3.00
Zilog 1991 Annual Report	DC-1991-AR	N/C
Microcontroller Quick Reference Folder	DC-5508-01	N/C